
μ COMP 80
POWERTRAN
μ COMP 80

A scientific computer — 1

Powerful design uses two microprocessors and BURP, a new high level language

by J. H. Adams, M Sc

This series of articles describe a complete scientific computer which is based on two microprocessors. Unlike conventional mini and microcomputers which rely on large and expensive memories for complex subroutines, the present design uses a Z80 to handle the general processing and leaves the number crunching to a MM57109 microprocessor. This second device contains all of the algorithms necessary to execute standard mathematical functions. The basic design uses 8K of memory although this can be easily expanded to 32K.

The series will also describe tests and diagnoses, together with the computer's operation, both in machine code and the high level language BURP. Games, mathematical and financial programmes will also be given, and the series will conclude with several options including graphics and symbolic displays, graph plotting and an e p r o m programmer.

A COMPUTER, to quote the dictionary, is an "apparatus for making calculations or controlling operations that are expressible in numerical or logical terms". With the advent of the microprocessor it has become possible to build low cost computers but, as industry has demanded, the majority of these devices are designed with the controlling-operations aspect of computing as their particular forte. Therefore, as numerical calculators, they are rather limited, and this has led to many disappointed owners of development kits who find their idea of microcomputers and that of the kit manufacturers are poles apart.

To meet the needs of such buyers, i.e. a system that can do sums and communicate in a language based upon English, various versions of the language BASIC (Beginner's All-purpose Symbolic Instruction Code, a programming language devised in 1964 by the Americans Kemeny and Kurtz) have been produced, ranging from Integer Basic, the mathematical applications of which are limited to computer games and the like, to quite complex versions which are able to handle floating point numbers and to perform several mathematical and trigonometrical operations on those numbers. These operating systems need computers with large memories, not for running large pro-

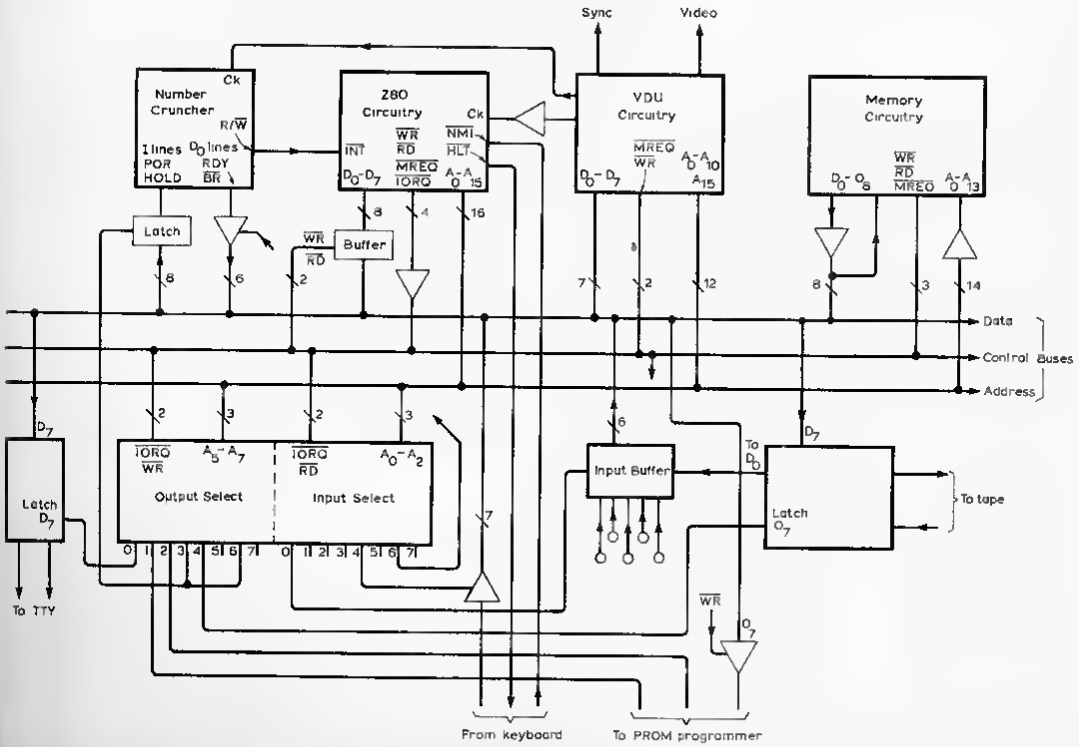


Fig. 1. *Block diagram of computer. Control and data buses are buffered as they leave the Z80, while buffering for the address bus is in the memory circuitry. Five buffered serial inputs and one buffered output are available for external add-on circuitry together with three output and five input select signals which may be used to latch data off or buffer it on to the data bus.*

Specification

Commands available

LOAD	Loads program lines into memory.
ADD	Adds program lines to those already in memory.
DEL n	Deletes program line n.
RUN n	Runs from program line n.
MOD*	Converts Print statements to Write and vice-versa. (For use with second output device.)
LIST n	Lists from program line n.
DUMP n	Lists on second output device.

Statements available

Input, Print, Write, For, Next, Goto or Go, If, Then, Gosub, Return, Top, Erase, Halt, Let.

Write is the print command for a second output device, such as a teleprinter.

Top clears the top line of the v.d.u. and sets this as the next printing position.

Erase is similar to Top except that the whole screen is cleared.

Halt stops execution until any key, except FS or RS is depressed.

Mathematical capability

Calculates to 8 figures plus 2 exponent digits.

Functions

+, −, ×, ÷, square, square root, log, 10^x, ln (nat log), e^x, sin, cos, tan, $\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$, y^x, $1/x$, π, degree to radian conversion and vice-versa.

Variables	26, denoted by A to Z.
Range	10^{-99} to 9×10^{99}.
Program lines	Up to 254, or the limit of the read/write memory.

Print capability

Automatic switching to scientific mode on results greater than 10^9 or less than 10^{-4}.

Printed figures may be tabulated or close-packed, to any number of decimal places from 0 to 7.

Automatic rounding occurs on results abbreviated in this way.

Alphanumeric data may be interposed with printed variables.

Input/output

Input via ASCII encoded keyboard.

Output via v.d.u. of 32 lines, 64 char/line. Separate video and sync signals to 625 standard.

Optional output to teleprinter.

300 baud f.s.k. input/output, using tones of about 1200Hz and 2400Hz, for the storage and retrieval of data via a tape recorder.

grams, but for storing the mass of information required to instruct the logic oriented microprocessor on how to behave as a number oriented device.

The aim of this project was to produce a computer with extended mathematical capabilities and avoid the need for such heavy investments in memory ics. This has been achieved by using two processors, the Z80 standard microprocessor which takes the dominant role as the processor of data moving around the system, and the MM57109, a number-oriented processor which handles the calculations. The Z80 has been well covered in this and other journals, but the MM57109 may be less familiar. This device appears to be a not entirely successful transplant of a scientific calculator chip into the world of data buses and memories. It can perform most of the standard scientific calculator functions and, in common with many such devices, it uses a sequence for instructions known as Reverse Polish Notation. This system differs considerably from the standard algebraic notation, and is based upon the logical idea that the instructions to be performed or, in the case of a calculator, keys to be pressed, should be listed in the order which they are to be performed. For example, consider the calculation $c = \sqrt{(3^2+4^2)}$. The first operator following the equals sign is the last operation to be carried out and yet the last, brackets, is not the first. The actual order of execution is in fact quite complex, and the more complex the expression to be solved, the worse things become. The algebraic sequence would actually be 3, sq, move to memory, 4, sq, +, memory recall, =, root. In algebraic BASIC, the computer line would be, LET C = (3↑2 + 4↑2) 0.5 where ↑ means raised to the power of.

However, a simple RP calculator would execute the operations in the sequence which the operator would follow, ie, 3, sq, store, 4, sq, recall, +, root. In practice this is even simpler because calculators and the MM57109 have a stack of registers, each capable of holding a number. In the MM57109, this stack consists of four registers called X, Y, Z and T for top. Data enters and leaves via the X register, but may be pushed up into the stack either for temporary storage, or to take part in a two number operation involving the contents of the X and Y registers (e.g. YX which calculates the Y number to the Xth power). There are specific instructions which move or exchange the contents of the stack registers to facilitate calculations, however, the system ensures that in normal use of the language, numbers are pushed or entered into the stack as and when necessary. As RP is a step-by-step system, no brackets are required, and in the expression originally considered the RP BASIC version of the computer line would be

LET C = 3 SQ 4 SQ + ROOT

Some other examples of calculations and the stack operations are given in table 1. In my view, having used both types of notation, the Reverse Polish wins every time. For this reason a new language was formed for the computer Basic Using Reverse Polish or BURP.

Hardware

The block diagram of the computer in Fig. 1 follows standard microcomputer techniques, with an eight wire data bus, a sixteen wire address bus and a four wire control bus which interconnect the various elements of the computer. The majority of lines in this design are active low, ie, for an i.c. output, they go to the low state when the particular output label is occurring, e.g. HALT on the Z80 goes low when it is in the HALT condition, WR goes low whenever the Z80 wants to write some information into the memory. With an input, that input must go low for the input label to occur, e.g. INT needs to go low for the Z80 to be interrupted. An active low label is identified by a bar over it.

To describe the operation of the c.p.u. shown in Fig. 2, it will be helpful if some aspects of microprocessor operation are discussed. The Z80 can execute a repertoire of 158 groups of instructions, of which there are about 600 in total, and these instructions are read in through the data bus of the system as 8-bit words or bytes. The reading in, or writing out of bytes from or to memory locations or input/output devices is controlled by the four processor output lines RD, WR, MREQ, IORQ. For example, a low RD and a low MREQ output from the Z80 indicates that it wants to read in a byte from a memory location, whereas a low WR and IORQ means that it is writing a byte out along the data bus to an output device, such as a teleprinter. The address of the memory location, which is stored in a 16-bit register within the Z80, known as the program counter, or code number of the input/output device, is simultaneously sent out onto a second 16-line bus by the Z80. External circuitry selects which memory location or device is coupled to the data bus for that particular Z80 operation. When pin 26 RESET, of the Z80 is taken to 0V, the program counter is cleared, and when pin 26 returns to 5V, the Z80 begins by reading in the data byte in memory location 0, executing it as an instruction, increasing the program counter by one, and reading in the next byte from memory and so on. Thus, the memory will contain lists of instructions to be executed sequentially, interspersed with bytes of data which are required by some of them. The Z80 will then work its way through these lists or programs.

The instructions cover such operations as LOADS, which move bytes

between registers within the Z80 or the memory. Logical and arithmetic functions, usually on the contents of the A register of the Z80. JUMPS, which, by feeding two new 8-bit bytes into the program counter register, cause the sequence of instruction execution to jump to a different point in the program. And CALLS, which are similar to jumps except that the old program counter contents are kept in a last-in first-out store in the read/write memory of the system, to be restored to the program counter on execution of the Z80 instruction RETURN. Calls are particularly useful whenever a certain block of instructions need to be used at several points in a program. If the instructions are written once in a program with a return instruction at the end, the block may be called at any point during the rest of the program. Such blocks are known as subroutines.

One feature of most microprocessors, including the Z80, is that CALL instructions may also be forced into the instruction sequence by activating either of the pins $\overline{NMI}$ or $\overline{INT}$. The subroutines called by these interrupts are executed immediately after the instruction in progress, and, as with most CALLS, once the subroutine has been completed, instruction execution recommences at the point where the sequence was originally interrupted. These interrupts are generally used by other devices that want to communicate with the Z80 and, in this case, the $\overline{NMI}$ interrupt is initiated by the strobe pulse from the keyboard and hence by the depression of any key. The keyboard subroutine reads in and acts upon the keyboard data before returning control to the main program as shown in Fig. 3. This is just one method of using the keyboard and another common approach is the polling system where, as part of the main program, the Z80 reads in the strobe pulse as part of a byte, then tests to see if the strobe is active and jumps back to the read operation if it is not. When it is active, the Z80 reads in the keyboard data byte. This method requires six bytes of instructions and memory locations, or at least a three byte CALL instruction to a read-the-keyboard subroutine whenever a byte of keyboard data is required.

In contrast, an interrupt driven system only requires the one-byte HALT instruction to be executed whenever a byte of keyboard data is needed. The only method for the processor to get out of the HALT state, once the instruction has been executed, is by operation of the reset button or by an interrupt, the Z80 waits for the interrupt which directs it into the subroutine for the keyboard. The interrupt system which was chosen, saves on memory space and the subroutine contains an extra section which will reset the entire system if the processor is not in the HALT state. If it is necessary to interrupt, for example, a program under development which has a fault, this can be achieved by pressing any key. The HALT command is also available in the high level language where it will also stop program execution until a key is pressed.

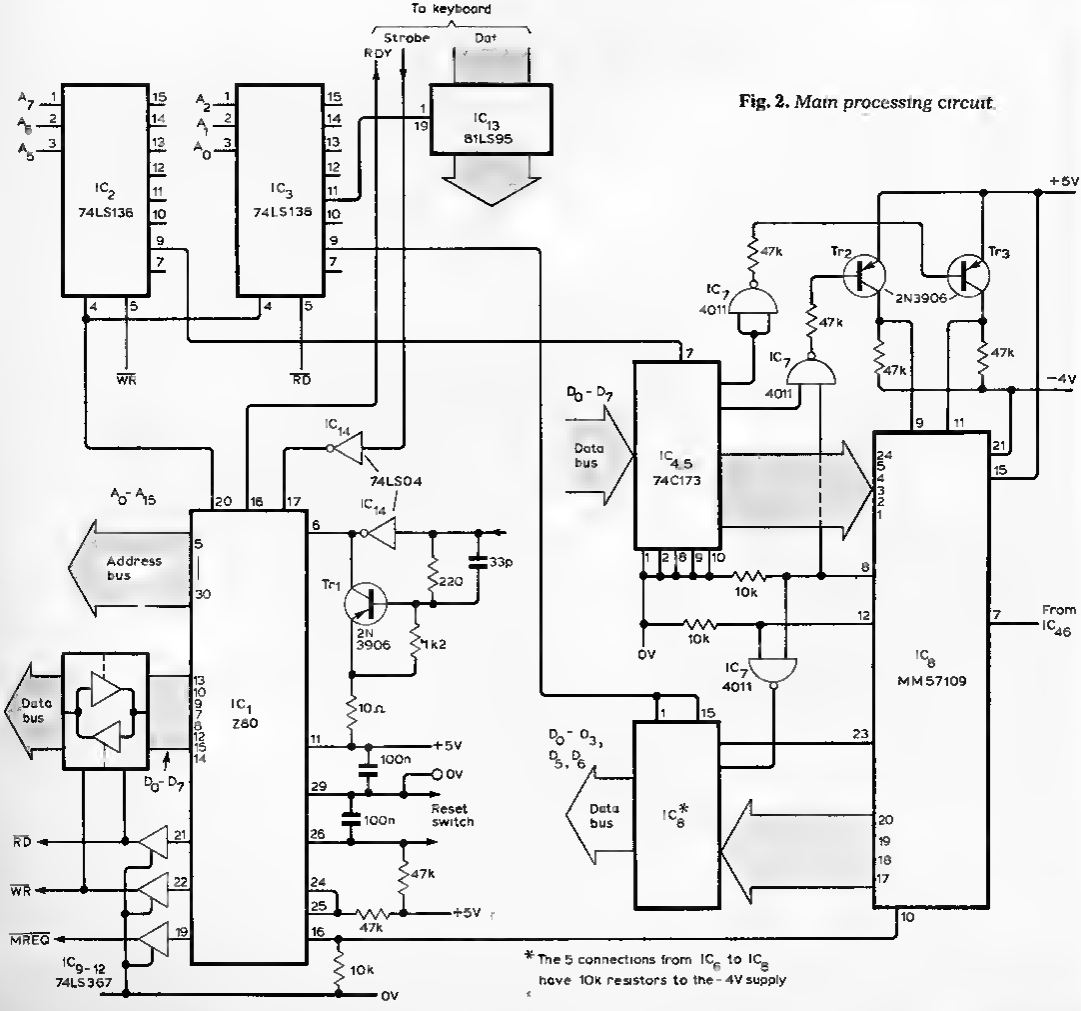


Fig. 2. *Main processing circuit.*

The NMI input (non-maskable interrupt) operates under all conditions and just calls to a fixed address in the memory. The other interrupt input, INT, can be enabled or disabled by instructions within the program being executed. This particular interrupt line comes from the MM57109 R/W output and indicates that the 57109 wants to send b.c.d. data to the Z80. This output can do some peculiar things during the initial reset of the processor, therefore, during this period, it is essential that the INT input is disabled. This is automatically done by the Z80 whenever it receives the reset signal. The interrupt can be programmed to respond in one of three ways, but in this system its response is similar to that of the NMI input, i.e., a CALL to a particular memory address.

Operation of the MM57109 is most easily understood by reference to the simplified diagram in Fig. 4. The device has open drain outputs which require external pull-down resistors, and some non-t.t.l. compatible inputs, but these have been left out of the diagram. The 57109 has a 6-bit input word into which 70 instructions, mostly of single bytes, have been encoded. The two relevant control lines are the RDY output and the HOLD input. RDY indicates that the device is ready to receive an instruction. If the HOLD input is low, RDY will go low after 16 μs, and the current instruction on the input lines will be executed. If the HOLD is high, RDY remains high and the operation of the 57109 is suspended until HOLD goes low. In this system, RDY is sensed, and HOLD is controlled by the Z80. The HOLD input is normally high, and the typical sequence used by the Z80 when it wants the 57109 to execute an instruction is shown in Fig 5.

Although this sequence is adequate for the execution of most instructions, the 57109 has been designed to operate as a separate microprocessor and will therefore sometimes produce RDY pulses during the execution of certain instructions. These pulses are intended to cue memory counters etc. As the HOLD is on during the execution of instructions, these RDY pulses must be supressed otherwise the Z80 may think that it's time for another instruction to be sent to the 57109 latch. Fortunately, in such cases the output ISEL at pin 12 goes low, and this is used to gate the RDY signal to, and the HOLD signals from, the Z80 via IC_7.

The data read in by the Z80 via the tri-state buffer, IC_8, consists of the modified RDY signal together with the four DO (digit output) lines which carry the b.c.d. data from the 57109 X register, and the BR line on pin 23. This line pulses low whenever one of the seven tests that the 57109 can perform proves to be true. Pin 10, R/W, goes low whenever b.c.d. data bytes are waiting to be read in by the Z80. During the execution of an OUT instruction, twelve such pulses occur which signal the two

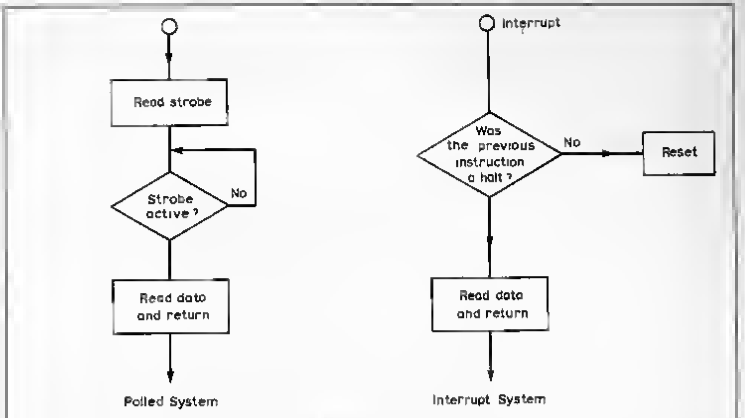


Fig. 3. *Polled and interrupt keyboard systems.*

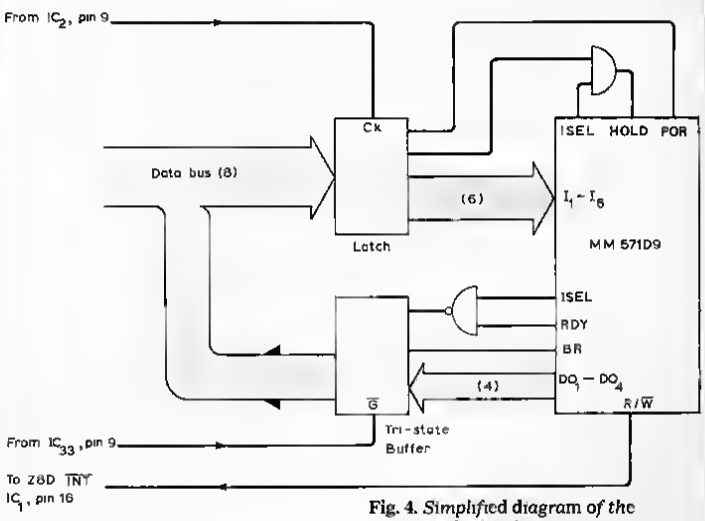


Fig. 4. *Simplified diagram of the MM57109 logic. The microprocessor also has open drain outputs which require external pull-down resistors.*

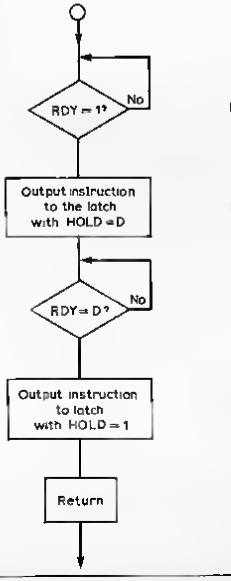


Fig. 5. *Typical sequence of Z80 operations when it requires the 57109 to execute an instruction.*

Table 1. Calculations which show the stack operations in the MM57109.

$\sqrt{3^2+4^2}$

COMMAND	X	Y	Z	T	
3	3 00	0 00	0 00	0 00	
SQ	9 00	0 00	0 00	0 00	
4	4 00	9 00	0 00	0 00	the stack is automatically pushed i e
SQ	16 00	9 00	0 00	0 00	X→Y, Y→Z, Z→T, T lost
+	25 00	0 00	0 00	0 00	the stack collapses, i e 0→T, T→Z
ROOT	5 00	0 00	0 00	0 00	Z→Y, result in X

$\frac{(6+9)}{(4-1)}$

6	6 00	0 00	0 00	0 00	
9	9 00	6 00	0 00	0 00	
+	15 00	0 00	0 00	0 00	
4	4 00	15 00	0 00	0 00	
1	1 00	4 00	15 00	0 00	
—	3 00	15 00	0 00	0 00	
/	5 00	0 00	0 00	0 00	

$\sin(1+\sqrt{(5^3-4)})$

5	5 00	0 00	0 00	0 00	
3	3 00	5 00	0 00	0 00	
YX	125 00	0 00	0 00	0 00	Y to the Xth power
4	4 00	125 0	0 00	0 00	
—	121 00	0 00	0 00	0 00	
ROOT	11 00	0 00	0 00	0 00	
1	1 00	11 00	0 00	0 00	
+	12 00	0 00	0 00	0 00	
SIN	0 2079116	0 00	0 00	0 00	

exponent digits, a byte representing the signs, one byte for the decimal point position, and the eight mantissa digits' readiness to be sent to the Z80. The interrupt caused by this line has already been described.

The HOLD and POR (power on reset) lines on pins 9 and 11 respectively, are not t t l compatible and Tr_2, Tr_3 in Fig 2 act as level shifters. Operation of POR occurs under the control of the Z80 whenever a reset is applied, and during the operation internal registers are cleared and various conditions within the 57109 initialised. It is during this operation that the R/W line goes low, but, as previously described, this is prevented from causing interrupts to the Z80. For further information on the MM57109, National Semiconductor produce a data booklet which gives full operational details of each instruction and pin function.

The clocks for the microprocessors are derived from IC_{30} and IC_{46} in the visual display circuitry. To meet the specified swing and rise times required by the Z80, a rise time of 30ns to a level of 4.4V, Tr_1 and the associated circuitry form an active pull-up on the output of the Schottky inverter in IC_{14}. The clock for the 57109 is obtained from pin 12 of IC_{46}. With the link from pin 1 of IC_{46} to pin 12 of IC_{23}, the frequency of this clock is 400kHz, which is the maximum specified in the data sheet. The other position of the link, to pin 14 of IC_{23}, doubles this frequency and, if the 57109 will operate at 800kHz as tested ones have, a worthwhile increase in computing speed can be achieved.

Tri-state buffers IC_9 to IC_{12} are connected to form an eight-line bus transceiver, and buffer the system control lines. These buffers also provide the extra drive required for the heavily loaded data bus. IC_2 is a 3 to 8-line decoder, activated by the control lines IORQ and WR. This allows eight different output devices to be addressed from the codes that the lower eight bits of the address bus holds during output operations. One of these output devices is the latch $IC_{4\,5}$. A similar job is done by IC_3 for input devices, and it is enabled by IORQ and RD. The three input lines to these devices are different, which spreads the load on the address bus. The input devices shown are tristate buffers IC_8 and IC_{13}, which buffer data from the 57109 and the keyboard respectively. □

To be continued

Literature Received

Digital optical **tachometer** is described by Compact Instruments Ltd in colour leaflet, available from Binary House, Park Road, Barnet, Herts EN5 5SA.

WW 401

Camera tubes. Plumbicon and Newvicon from Mullard are on colourful, but not very informative, wall chart which can be obtained from Department C1H, Mullard Ltd, Mullard House, Torrington Place, London WC1E 8HD.

WW 402

A range of alphanumeric **printers** is subject of brochure recently sent to us by Syntest, 169 Millham Street, Marlborough, Mass 01752, U.S.A.

WW 403

General and specific information on **stepping motors** is given by Moore Reed in their 50 page catalogue, which can be had from Moore Reed and Company Ltd, Walworth, Andover, Hants SP10 5AB.

WW 404

Computer **card recorders** for analogue or other information are made by Houston Instrument and described in a brochure from Louis Arnold, Houston Instrument, One Houston Square, Austin, Texas 72753, U.S.A.

WW 405

Brief descriptions of test and measuring **instruments** marketed by Lyons Instruments in product guide, not sent to us but available from L.I. at Ware Road, Hoddesdon, Herts EN11 9DX.

WW 406

Audio test instruments produced by Bang and Olufsen shortly described in recent brochure, obtainable from Eastbrook Road, Gloucester GL4 7DE.

WW 407

Health and safety recommendations for Bakelite materials in two publications. TIS B212 for impregnated materials, TIS B211 for industrial laminates. Copies from Bakelite UK Ltd, Sales Office, Tuition House, St George's Road, Wimbledon, London SW19 4DS.

WW 408

Sound reinforcement systems made by Millbank, described by Royal Institute of British Architects in a product data sheet, is available from Millbank Electronics Group, Uckfield, Sussex TN22 1PS.

WW 409

Company magazine of Rohde and Schwarz for end of 1978 deals with range of r.f. test gear and television techniques. Includes description of standard stereo decoder. In English. Rohde und Schwarz, Postfach 80 1469, D-8000 Munich 80, Fed. Rep. Germany.

WW 410

Catalogue of voltage and current **stabilizers** is produced by Techmation Ltd. Units are made by Kepco and use switching techniques, ferroresonance and ordinary feedback methods. Techmation Ltd, 58 Edgware Way, Edgware, Middx.

WW 411

A scientific computer — 2

Memory, v.d.u., tape and teleprinter interface

by J. H. Adams, M Sc

THE STANDARD COMPUTER holds up to 8 kilobytes of memory, but this can be expanded up to 32 kilobytes if necessary. Of this memory, 3K are 2708 r.o.ms which are mapped, or located, at the hexadecimal addresses 0000 to 0BFF. The first 2708 contains a monitor program using hexadecimal data, and responds to commands entered in English. The remaining two r.o.ms hold the more complex BURP monitor and interpreter.

Addresses 0C00 to 1FFF accommodate up to 5K of 21L02 read/write memory and these i.cs are grouped in eights, each one being the store for one bit in an 8-bit byte as shown in Fig. 6. The computer will operate with only 2K of r/w m., these being IC_{22} and IC_{26}, but, for greater program storage, further blocks starting with IC_{23} should be added. Pin 12 of IC_{17} should be wired to pin 13 of IC_{22} and the position for IC_{21} left blank. This has been provided for any future development which may require more r.o.m. space.

The data outputs of IC_{18} to IC_{26} are wired to an eight-line bus which feeds into the main data bus through the tri-state buffer IC_{27}. Parallel wiring of the outputs is possible because the memory devices also have tri-state outputs. The chip-enable, CE, inputs are enabled by the outputs of a 3 to 8-line decoder, IC_{17}, which is, itself, only selected when $\overline{MREQ}$ and address line I5 are low, i.e. the Z80 must be requesting a memory operation in the lower half of the memory. Similarly, IC_{27} is only enabled when both $\overline{MREQ}$ and $\overline{RD}$ are low, i.e. a memory read is requested. A link has been placed in the $\overline{MREQ}$ line so that the system may be expanded as shown in Fig. 7. The $\overline{MREQ}$ is connected to a 74LS139 2 to 4-line decoder which is fed by A_{13}, A_{14} and the original $\overline{MREQ}$. This modification generates four new $\overline{MREQ}$ signals, one for each quarter of a 32K memory. If the expanded memory is used, $\overline{MREQ_1}$ will take the place of $\overline{MREQ}$ on the original board.

As there are many devices to be

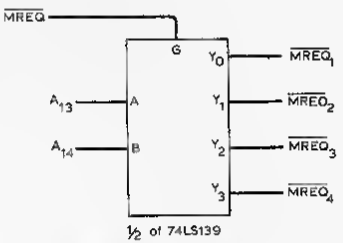
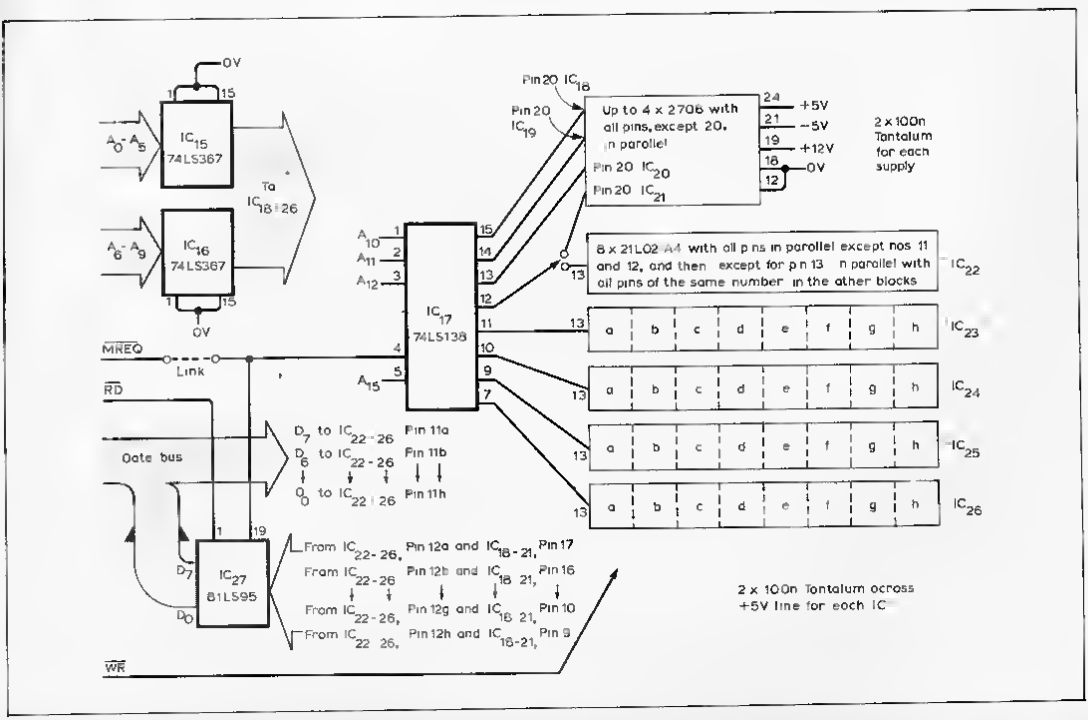


Fig. 7. *Modification to achieve four $\overline{MREQ}$ lines for the control of up to 32K of memory.*

Fig. 6. *Memory circuit. 3K of r.o.m. contains the monitor program, BURP monitor, and interpreter. 5K of r.a.m. accommodates user program storage. The memory can be extended in blocks of 8K up to 32K.*



driven from A_0 to A_9, these lines are buffered by IC_{15} and IC_{16}, which are permanently enabled. Although adding buffering increases the time delay between a memory request, and the data being ready for the Z80 to read in, the most critical 'reads' have 750ns to access the memory, therefore 21L02s with a 450ns access time are quite adequate

Visual display

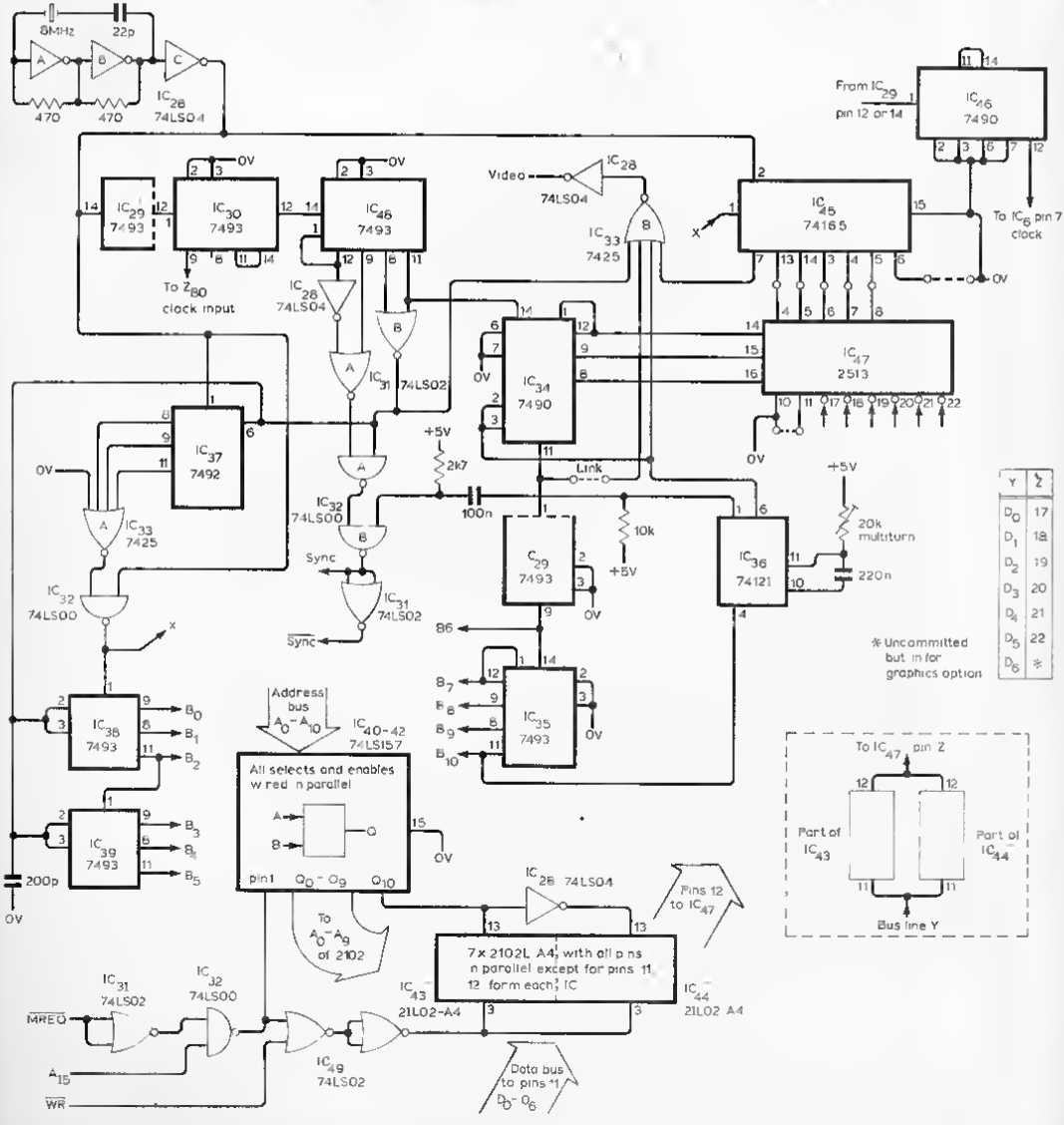
The visual display circuitry in Fig. 8 is constructed using standard t.t.l. i.cs instead of one of the recent l.s.i. controllers This approach was chosen because the only suitable l.s.i. device is rather expensive, and, more important, the t.t.l design gives maximum flexibility and allows an optional graphics system to be easily implemented

Each line scan takes 64μs and is thus compatible with standard 625 signals and, with 320 instead of 312.5 lines per picture frame, the frame scan, although slightly longer, will easily be within the range of a television set to be used as a v.d.u

In Fig. 8., IC_{28} generates a signal at 8MHz from which all of the clocks within the computer are obtained. Part of IC_{29} divides the clock to 4MHz, and either the input or output of this i.c. is fed to IC_{46} which produces a true square wave at either 800 or 400kHz for the MM57109. The 4MHz is further divided by IC_{30} to provide a 2MHz clock for the Z80, and an output at 250kHz for IC_{48},

Fig. 8. *Visual display circuitry This discrete t.t.l version reduces cost and improves flexibility*

which produces 16 states of 4μs each State 0001 is decoded and used as the line synchronizing pulse The output at pin 11 of this i.c. is at 15.625kHz, and clocks IC_{34} at the standard 625-line rate Part of the line sync. decoder, IC_{31b}, provides a signal which is active low during states 0100 to 1111 inclusive This produces a 48μs burst during which IC_{33} and IC_{37} to IC_{39} are enabled, and allows a video display to occur, see Fig. 9. Division of the 64μs line scan into a 4μs sync. pulse, an 8μs pause, a 48μs display, and then a 4μs pause, allows for the overscan that occurs in commercial televisions. If the displayed video runs off the end of the screen, even with the width and shift controls adjusted correctly, omitting the inverter on pin 12 of IC_{46} will shift the displayed video 4μs to the left which corresponds to about five characters.



The dividers described so far are free-running with their reset inputs taken low. The decade counter IC_{34} is fed with pulses at the line frequency and divides these to provide 10 row addresses for each line of displayed characters. Outputs Q_1 to Q_3 go directly to the character generator, IC_{47}, but Q_4 is connected to IC_{33} and is high during the ninth and tenth lines, i.e. the two lines to be blanked between character rows. A divide-by-32 counter, formed by IC_{29} and IC_{35}, provides an address for each line of characters on the screen. At the end of a display page, monostable IC_{36} is triggered by this counter to provide a reset pulse for IC_{34}. This holds the divider chain, IC_{34} to IC_{39}, at the first row of the first character of the first character row until frame flyback has occurred and line sync has been regained. The monostable pulse is also used to blank the video during the flyback via IC_{33}, and through a differentiator, to provide a field synchronising pulse to IC_{32}.

A third divider chain remains reset except during the 48μs display time defined by IC_{48} and its associated gates. IC_{37} is clocked at 8MHz, and divides-by-6 to provide a character-rate pulse, which, when gated with the 8MHz clock, is used to load data from the character generator into the shift register IC_{45}. This gated pulse is also used to clock a character counter divide by 64 circuit, comprising IC_{38} and IC_{39}. Data loaded into IC_{46} comprises five bits of the row of the character which the character generator is being asked to supply, together with one blank bit on pin 6 to complete the 10 × 6 character format. These bits are loaded in parallel and clocked out serially, via pin 7, to the video gate, IC_{33}. The three lower address bits for the character generator, IC_{47}, are supplied by IC_{34} as previously explained. The upper six come from the r/w m. comprising IC_{43} and IC_{44}. Addresses for this block of r/w m are supplied by the 2 to 1-line multiplexers IC_{40} to IC_{42}, which are switched between addresses supplied by the main address bus of the Z80 and those being generated by the v d u dividers by the decoded $\overline{MREQ}$, $\overline{WR}$ and A_{15} signals. Because this signal also switches the $R/\overline{W}$ inputs of the memory, the conditions are either; multiplexer connects Z80 address bus to r/w m., Z80 writes into r/w m, or multiplexer connects v d u. circuitry to r/w m, data inputs to r/w m disabled.

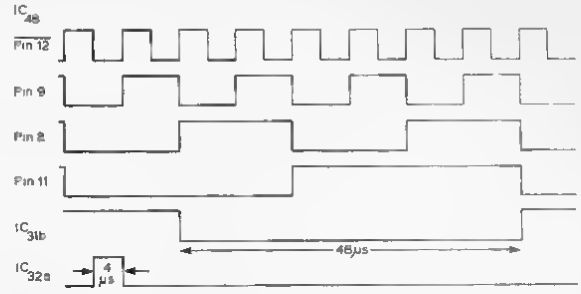


Fig. 9. *Divisions of the 64μs line scan*

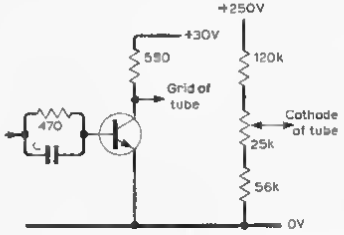


Fig. 10. *Typical television interface*

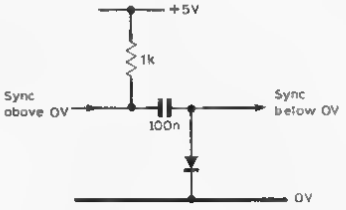


Fig. 11. *Positive to negative sync. converter*

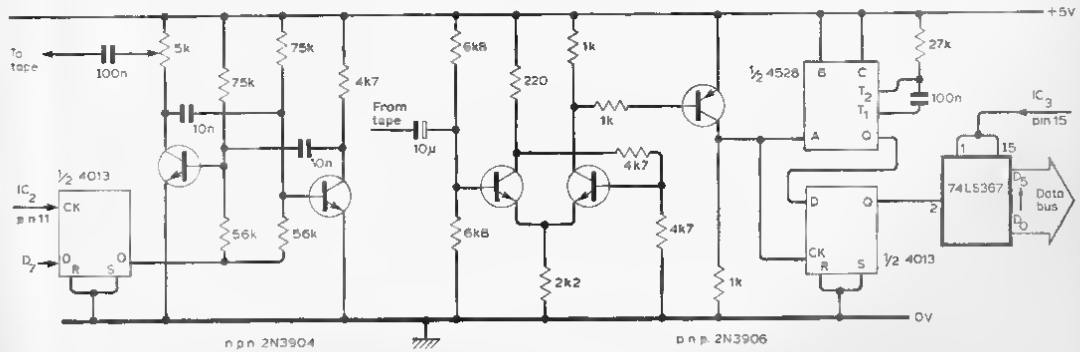
Fig. 12. *Tape interface. The receiver is not susceptible to changes in tape speed because it does not detect a particular frequency, but recognises whether the signal is above or below a certain frequency*

Each i c block in the r/w m consists of seven 2IL02A4 devices, six of which are used as stores for ASC11 character codes. The seventh is for an extension of the v d u to graphical displays and may be omitted. The 2 kilobytes of memory contained in the v.d u is mapped at the addresses 8000 to 87FF, but, due to the simple decoding used to address the v d u., it may also be "found" at 8800 to 8FFF, 9000 to 97FF etc, up to F800 to FFFF, i.e. 16 blocks in all. This provides a simple method of extending the v d u. display so that, after printing out a full screen of results, the next line will automatically start at 8800, i.e. the top of the screen, and so on, which effectively gives over 500 lines of display.

Tv interfacing

The v d u circuit produces approximately 4V of video information and a separate 4V synchronization signal consisting of 4μs and, approximately, 100μs pulses. If a dedicated monitor is not available, perfectly good results can be obtained with a modified 625 line television as follows.

To obtain a high quality display, the video amplifier in the tv set should be ignored, and the v.d u. signal applied, via a buffer, directly onto the grid of the tube. As the chassis of most sets is connected to one side of the mains, an isolation transformer must be placed between the mains supply and the tv/



computer. Apart from the danger to yourself if this is not done, there is a great risk of destroying several ics with one mistake.

A typical interface for a tv set is shown in Fig. 10. If a 30V supply is not available in the set it can be obtained, via a diode and smoothing circuit, from the cathode of the line output stage, or from the computer. The interface should be mounted as close as is possible to the tube base, and the video fed via good quality r.f. coaxial cable from the computer. Capacitor C is selected to compensate for the particular length of coax used, 50pF being a typical value for 1 to 2m of miniature cable.

The best place to inject a sync signal is where the sync. is taken from the i.f./video section to the timebase. An oscilloscope should be used, with the set operating normally, to identify where the voltage level of the sync pulses best matches that of the computer. When this has been ascertained it is important to note whether the pulses are positive or negative going, and whether, as is common in valve sets, the entire sync signal is negative. The first point will determine which sync. output is used, and Fig. 11 shows a simple method of dropping the entire sync. signal below 0V.

Tape interface

It is useful to have a cheap means of storing data and, at present, recording on ordinary cassette tape is the most acceptable method. Blocks of data are recorded onto, or read off tape via the interface shown in Fig. 12.

Data is recorded at 300 baud, with 0s and 1s represented by tones at approximately 1200 and 2400Hz. The Z80 calls the byte to be recorded into register A and sends it out, bit 7 first, along data bus line D_7. The byte is prefixed by a 0, known as a start bit, and immediately followed by 1½ bits, in duration, of a 1, which is known as a stop bit, see Fig. 13. Thus, the recorded word is 10½ bits long, i.e. 35ms. A time delay is generated by the Z80 (its fastest rate being over 100 Kilobaud) and latches the bits from line D_7 into the D type flip-flop, whose Q output switches a multivibrator to generate the tones.

The receiver consists of a Schmitt trigger and buffer, which produce a 5V pk-to-pk square wave from the tape deck output, followed by a frequency discriminator formed by a monostable and the other half of the D type flip-flop. The square wave at 1200 or 2400Hz is simultaneously used to trigger the monostable, and clock its output into the flip-flop. The pulse width of the monostable has been chosen so that with a triggering rate of 2400Hz, its output is continuously at 1, i.e. a pulse width > 1/2400s. Therefore, the output of the flip-flop is also at 1. With a triggering rate of 1200Hz, the monostable has time to complete its pulse, so the next clocking pulse clocks a 0 into the flip-flop, i.e. a pulse width < 1/1200s. The circuit requires a 2 to 3V pk-to-pk signal from the tape deck.

The firmware starts each recording with several seconds of stop bits before the transmission of data begins, and this allows the tape to be cued before a READ command is given to the computer. The read software operates as shown in Fig. 14.

The bit from the interface is read in through a 6-bit buffer and then masked, i.e. the other bits are blanked off by a Z80 AND instruction, before checking

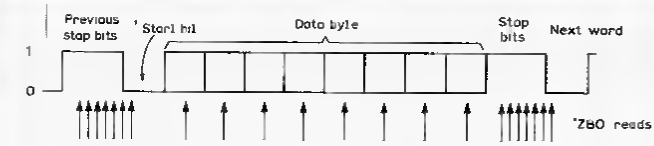


Fig. 13. *Word structure of a software recording.*

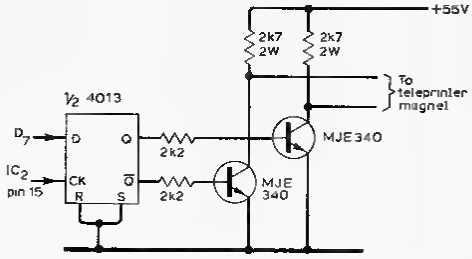


Fig. 15. *Teleprinter interface. This circuit is kept simple because software in the r.o.m. carries out most of the operations.*

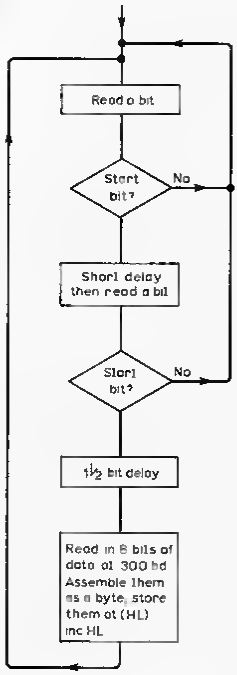


Fig. 14. *Operation of the read software.*

takes place. To give error protection, two checks must indicate that a 0 is being read in before the routine proceeds, and a byte of data is then read in.

Teleprinter interface

A teleprinter is a very useful addition to any computing system, but, in a low cost design such as this, a new machine would be prohibitively expensive. The supplies of second-hand machines seem to split into two groups; 7/8-bit machines using ASCII coding, and 5-bit machines using modified ASCII or Baudot coding. The advantage of the first type is that they hold the full ASCII set of characters but, they are still reasonably expensive. The 5-bit, modified ASCII machines, such as the Creed 75s, can be purchased for as little as £5, but only have 32 possible codes. Almost 60 different characters can be generated by using two of the code words, 1B and 1F, to switch a mechanical flip-flop, within the teleprinter, between two sets of characters, letters or figures and punctuation. The two shift characters, line feed, carriage return, blank and space remain common to both shifts.

This type of machine requires a serial input at 75 baud which means that, with a 7½-bit word, it will print 10 characters per second. Coding for the letters is identical to the lower 5 bits of ASCII code, but the figures, although similar, require translating from the 6-bit subset of ASCII used in the computer, to the 5-bit code for the teleprinter. The formation of words comprising a start bit, five data bits and two stop bits, the interspersing of figure and letter shifts where necessary, the storage of a byte which mirrors the state of the mechanical shift flip-flop in the teleprinter so that shift characters are only

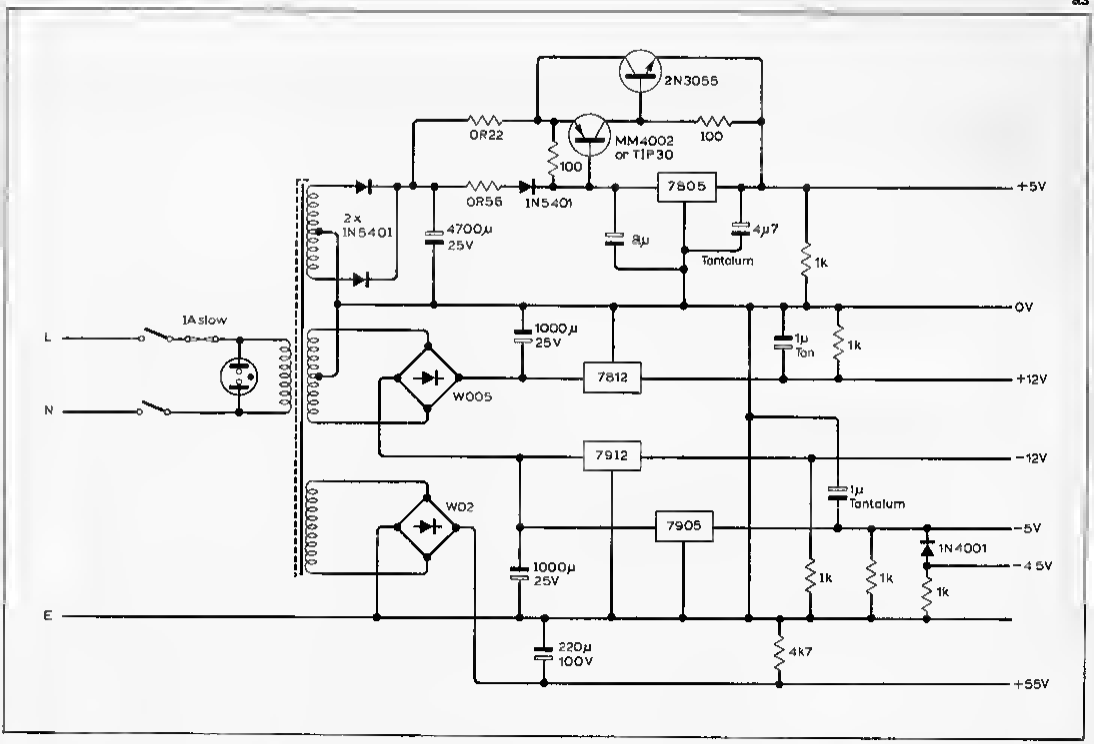


Fig. 16. *Power supply. A 1A regulator in parallel with a general purpose power transistor gives a regulated output of up to 3·5A*

sent when necessary, and the time delays required to slow the computer down to 75 baud, are all accomplished by software stored in part of the monitor r o m. The hardware involved in interfacing the teleprinter, Fig. 15, is therefore very simple.

Power supplies

The power supply provides stabilized outputs of +5V at 3·5A, −5V and ±12V at 1A, and an unstabilized output of approximately + 55V at 100mA. The +5V supply in Fig. 16 is the only unusual part of the circuit as it uses a 1A regulator i.c. with a pseudo p-n-p pass transistor instead of a more expensive 3A regulator. The circuit operates by dividing the current passing through the regulator and the transistor in the ratio of the two series resistors. In this case, 56/22 times as much current passes through the 2N3055, which gives a 3·5A regulated output for 1A through the 7805. It is important that the regulator, transistors, and the diode which mirrors the voltage drop across the base-emitter junction of the p-n-p transistor are in good thermal contact.

The −4.4V supply is for the MM57109 which is specified to operate from a maximum overall supply of 9.5V. The +55V supply is for the teleprinter magnet. Although the Creed magnets, in common with others, are designed to operate with a current of 20mA, and have a resistance of 200Ω, switching 4V into the coil is not satisfactory due to the inductance in the magnet's coil.

Therefore, current driving, rather than voltage supplying, is required to produce a quick action from the magnets, and a high voltage supply fed through 2·7kΩ resistors is used. If a teleprinter is not going to be used, it is still worth including this supply as it may be needed for the tv interface, and will be required for the r o m programmer to be described later.

The transformer used in the prototype was an RS Components 50VA d.i.y. type 207-554 with windings to give open-circuit voltages of 9.5-0-9.5V for the +5V supply, 13V for the −5V and ±12V supply, and 40V for the teleprinter supply, the last mentioned being wound with 36 s.w.g. wire and the other two with 22 s.w.g. wire.

No filtering has been included at the mains input as the processor is relatively insensitive to interference from the mains cable. The computer has been used to receive radioteleprinter messages via a communications receiver with an interface, and tests showed that negligible interference leaves the computer by the mains cable.

To be continued

A kit of parts for constructing a computer on this design is available from Powertran Computers, Portway Industrial Estate, Andover, Hants SP10 3NN (tel. Andover (0264) 64455).

The author

John Adams read for a B.Sc. in electronic engineering and for an M.Sc. in materials science at the University College of North Wales, before becoming a teacher. His involvement in electronics is, therefore, primarily as an amateur, and the design for this computer started 18 months ago as an exercise to gain familiarity with microprocessors. John operates the amateur radio station G3VZF, and in his spare time claims to be a very amateur musician.

A scientific computer — 3

Construction, testing and operating

by J. H. Adams, M.Sc

ALTHOUGH this is not a simple project, with careful soldering and the usual m.o.s. precautions, the construction should be quite straightforward. It is worthwhile building the power supplies first and testing them under load conditions of 3A for the +5V and 0.5A each for the −5V and +12V, until the regulators have reached their working temperatures. As a power supply failure can be particularly damaging, a generous heatsink, especially on the 2N3055, is recommended.

The next section to build should be the v.d.u. circuit, which will provide the video and sync signals required in the development of the display interface. To ease later work, the interface should be built as described in part 2. With the character generator and the 21L02s left out, and with the variable resistor set to a maximum, a correct display will consist of 32 rows of 64 oblongs. With the character generator and memories in place, these oblongs will become rows of random ASCII characters. When this is displayed, the variable resistor is reduced to move the display up the screen until it is as high as possible with correct linearity of all 32 lines. Reducing the resistor too much will either cramp or expand the top line and eventually wrap it back into what will then become visible fly-back. Table 2 gives test points and their waveforms for the v.d.u., and table 3 gives processor checks.

Once the circuit has been completed it should be thoroughly checked. A particularly devastating fault occurs if power lines appear on the wrong i.c. pins, especially the outputs of t.t.l. circuits. An ohm-meter, connected between each of the supplies in turn and the i.c. pins, will check for this kind of fault. Cautious constructors need only insert IC_{18} and IC_{26} out of the memory devices, the first r.o.m. and the r/w.m. covering 1C00 to 1FFF, for the initial test. Fig. 17 gives a suitable sequence for these tests.

The computer requires an ASCII coded input, comprising 7 bits of inverted data, together with a positive strobe pulse, active during the presence of the code at the computer input buffer. The Carter type 756 keyboard will give such signals when connected as shown in Fig. 18. For those constructing a purpose-built keyboard, DEL, ESC, CTRL and — are not required, and RS should carry the legend ↗S the legend ↓. The l.e.d. lights whenever the Z80 is in the halt state and indicates that the computer is waiting for keyboard data.

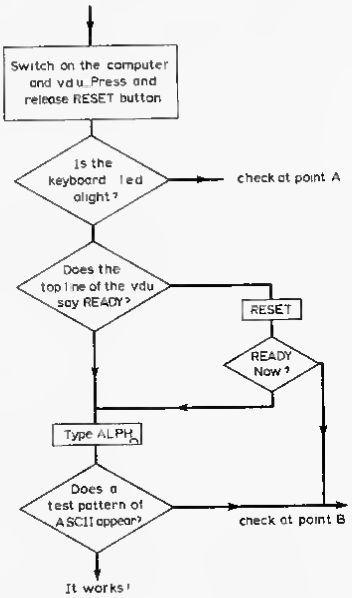


Fig. 17. *Test sequence for the computer*

Fig. 18. *Connection for the Carter 756 keyboard*

Using the computer

When the assembled computer has been tested, the r.o.ms and at least IC blocks 22 and 26 should be inserted. Two programs, one in the low-level and one in the high-level language, are programmed into some spare space at the end of the third r.o.m., and these will be used to demonstrate the computer's ways.

In the tables and explanations of commands and program lines, the symbol ~ means that a space has to be typed at that point, e.g.

TAPE ~ 1800 1880

means that you type TAPE space 1800 1880. As explained earlier, this is one of the bases of the systems operation.

Low level operation

When you have a high-level language, working in machine code may seem like talking in Morse code. However, low-level programs, if properly written, usually occupy less memory space, run faster and allow the computer to be used as a controller of processes, as well as a calculating machine. Table 4 lists the computer's machine code commands. At the address 0B16, there is an example of a code-breaking game where the computer makes up a four digit number using the digits 1 to 8, and then marks your attempts to guess the code by awarding black symbols for correct digits in the correct place and white symbols for any remaining digits in the code which are in the wrong place. With

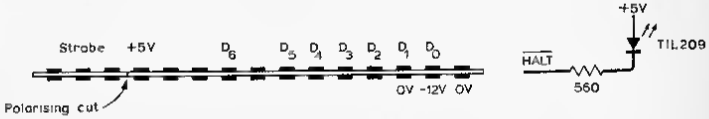


Table 2 v.d.u. test points and waveforms

Location	Waveform	Possible remedy
IC_{28c} out	8MHz clock, t.t.l.	IC_{29} or crystal
IC_{32a} out	4μs pulses every 64μs	check back through IC_{46}, IC_{30}, IC_{29}
IC_{35} pin 11	approx 50Hz, t.t.l.	check through IC_{35}, IC_{29}, IC_{34}, and then
IC_{36} pins 1–6	approx 100μs pulses every 20ms	check IC_{36} pin 6 is normally low
IC_{37} pin 2	16μs pulses every 64μs	check IC_{31b}, IC_{48}
IC_{45} pin 1	1.333MHz with 8MHz bursts	check IC_{37}, IC_{33}
IC_{45} pin 7	48μs bursts of data every 64μs	check IC_{45}
Video	mixed video and blanking information	check IC_{33}
Sync	mixed syncs	check IC_{32b} and the differentiating network

the computer in the READY state, type

RUN . 0B16

and then your first guess at the code, say 1234. The computer will mark your guess and wait for your next attempt. Note that, as soon as you type something, in this case, the first letter of RUN, the READY disappears, and does not return until you break the code, indicating that the program has finished running. The READY state may be achieved at any time by pressing RESET, or by typing FS. To examine the code set during the program, return to the READY state and type

LIST . 1FE4

The computer will then list from address 1FE4 to 21CF. The format used for listing gives the address of the first byte, which will appear on that line, and then the remainder up to the end of the row of 16, spaced in blocks of four for easy inspection. When a line is broken into, as in this case, the computer maintains the layout by indenting the top line by the correct amount. The first four bytes contain the computer's code, a 00 representing a digit 8.

The game may be played over and over again, using the command

RUN . 0B16

but, as an illustration, suppose that the program is to be simplified. To alter the program, it must first be copied into the r/w m so type

MOV . 0B16 0C00 1E16

which will move the program out into r/w m and, because some of the bytes in the program relate to the memory area that the program occupies, type

COR . 1E16 1F00 0B 1E

The computer will reply

1E24 1E37 1E3D 1E5B 1E62

meaning that it found 0Bs at these addresses, and changed them to 1Es. Now list the program,

LIST . 1E16

and note that the byte at 1EAC, the 13th byte on the row starting 1EA0, is a 77. Type

ALT . 1EAC 33

at which the 77 will change to a 33. This will limit the number range in the code from 1 to 4, rather than 1 to 8 as in the original. The computer will not return to the READY state because often more than one modification is carried out at a time and, as in this case, the 0B at 1E5B, which was altered in the COR command, was not part of an address to be altered, but is the k in the word black, and so must be changed back with

1E5B 0B

Now, press FS to achieve the READY state and

RUN . 1E16

to play the simpler form of game.

Using the machine code is essentially a matter of practice and experience, but more details will appear in part 4.

High level language

Table 5 lists the BURP statements, and

Table 3. Processor checks.

Location	Waveform	Possible remedy
Point A IC_1 pin 18	low	If it is low test the l e d
Point B IC_1 pin 6	8MHz	If not check clock buffer circuit around IC_{14}
Address bus	Various	A_0 to A_6 should be cycling through refresh addresses A_7 to A_{15} should be low except for A_8, which carries a 500kHz square wave. A line not conforming to this pattern is not necessarily at fault. Check for levels between 0.8 and 2.4V as these imply a short to one of the other address lines
$\overline{RD}$	750ns pulses every 2µs	These are active low and so the pulses described are to a low state. Repeat the test for shorts
$\overline{WR}$	High	
$\overline{MREQ}$	750 then 500ns pulses every 2µs	
$\overline{IORQ}$	High	
Data bus	Various	During the $\overline{RD}$ pulse the computer accesses the memory although as it is the HALT state (when working correctly) the Z80 ignores the accessed byte. The accessed address is 0357, which puts the byte $E6_H$ or 11100110 on lines D_7 to D_0 respectively during the pulse. If these lines are tested, short circuits must not be confused with tri-state periods, when the lines may float into the intermediate voltage range

Table 4. Machine code commands.

ALPH .	Produces an alphanumeric test pattern on the v d u
*ALT . XXXX YY	Changes the contents of location XXXX to YY
COR . XXXX YYYY AA BB	Scans XXXX to YYYY-1 inclusive and alters any AA to BB
FILL . XXXX	See note 3
FIND . XX YY	Finds the consecutive bytes XX YY and lists the addresses at which they occur
LIST . XXXX	Lists the contents of the memory from address XXXX up to a full v.d.u. screen
LOAD . XXXX	Loads hexadecimal data at XXXX using the same display format as in list. To load ASCII directly type a [and to return type a] after which the computer gives the next byte's address and continues to load hexadecimal data. To leave LOAD, type @ which gives a full listing of what has just been loaded or press RESET or the FS key to regain command
MOV . XXXX YYYY ZZZZ	Moves the block XXXX to YYYY-1 inclusive to the area of memory beginning with the address ZZZZ.
PRINT . XXXX	Lists from XXXX on the second output device
PROM .	Used in conjunction with the e.p.r.o.m. programmer this programs the block of data at 1C00 to 1CFF inclusive into the sector of the 2708 selected on the programmer
READ . XXXX	Reads from tape into memory starting at location XXXX. READ must be terminated by pressing any key once the tape has been read in
RUN . XXXX	Runs from address XXXX
TAPE . XXXX YYYY	Records, on tape, a short leader of stop bits, followed by the data at locations XXXX to YYYY-1 inclusive and a short trailer of stop bits

*Having accepted the alteration, the computer lists out from the previous LIST command starting address. Normally a LIST of the area in which the alteration is to take place will have been carried out immediately prior to using ALT, and so the altered byte will change to its correct contents on the v.d.u. screen as well as in the memory. After an alteration the computer does not return to the command state, but waits for any further alterations typed in as

XXXX YY

and so on. To return to the command state type FS or press RESET.

Notes

1. Take care when using MOV. MOV 1D00 1E00 1CFF will work, and move the block 1D00 to 1DFF inclusive, forward one byte in memory, but MOV . 1000 1E00 1D01 will copy 1D00 into 1D01, then 1001 into 1002 etc., leaving you with a block of identical characters and your original data lost. While this can sometimes be useful for filling out a block with a particular byte, to do this properly requires a MOV of the block to a separate vacant area and then a MOV to 1001.
2. The PROM . command takes about 40s to program the e.p.r.o.m. sector completely, and during this time the computer is fully occupied.
3. If less than 256 bytes are to be programmed into an e.p.r.o.m. sector, and the other must be left blank for later additions to the e.p.r.o.ms contents, or, if you wish to add this later bit to an already partly filled e.p.r.o.m. FFs must be present at the bytes which are not to be programmed. This can be achieved by using the extra command FIL . XXXX, which will fill from XXXX to the next YY00 with

FFs, either before loading into the p r o m eree, or after loading to mask off the other unused bytes up to 1CFF This commend also mokes programs easier to study on the v d u es it can be used to mask off the rubbish following a program.

4 When loading ASCII, do not try to include e] in the string of characters es it will terminate your ASCII mode of loading Also do not type in any further [es these will be used in a future edaption to graphics, whose firmware is already in the 2708 Ordinary perenthesas (and) are quite eccetable to the computar

Table 5. Burp statemente.

INPUT . A . B . etc	Inputs and essigns one or mora vanables
LET . X=A . SIN . SQ . etc	Assigns the velue computad in tha expression following the = sign to X
IF . X=Y . THEN . 50 .	If the condition (which may be <, = or >) is met than go to-line 50 Otherwise continue
FOR . X=1 . STEP . B . UNTIL . C .	X takes the velue 1 the lines up to the line NEXT X era than executed X is then incraasad by B and the lines executed agein and this continues until X is greatar or equal to C et which point the computer carries on through the line NEXT X to the next one
NEXT . X .	
GOSUB . 200 .	Goes to line 200 and executes from thera until the line RETURN is found, and then returns
RETURN .	to the line following GOSUB GOSUBs mey appeer within GOSUB blocks
HALT .	Halts execution until eny key is pressed
TOP .	Clears end resets tha PRINT position to tha top line of the displey area.
ERASE .	As TOP but it clears the whole display erea
END .	Stops execution end returns to the command stete
GO . 25 . or GOTO . 25 .	Executes from line 25
WRITE . . .	As PRINT for the second output device
PRINT . . .	
In prints the following may eppear	
An .	A printed with n figures aftar the decimal point end then spaces for the blenked characters 13 in all For less then an 8 digit mentisse, the last figure is rounded if necessary
A .	A printed with the seme number of figures efter the decimal point es the previously printed veriable or if it is the first one to be printed to four figures This four can be altered in r o m location 0818 by programming 01 to 07 in place of the 04
An .	As An . but with tha blanked figures completely suppressed and if the number is in scientific notation with tha exponent and exponent sign egeinst the mantissa
A .	As A . but with the suppression described above

Summarising without the comma printed figures elways occupy 13 screen locations and thus columns of results will be tabulated no matter what the magnitude of the number With the comma, elphanumenc deta (see below) end veriables may be printed in tha sema line without larga gaps appearing

"PRINTED TEXT" .	Prints the actual characters within the quotes implying that quotes must not appear in the string of charecters

It is not possible to have e second (or subsequent) FOR NEXT block within a FOR NEXT block because the single on-chip memory in tha MM57109 is used es a loop counter in conjunction with the NEXT line These loops if required can be set up using for examplo in place of the FOR line given,

```
20 . LET . X = 1 .          } replaces the FOR
21 .                        }
22 .                        } lines within FOR end NEXT
23 .                        }
24 . LET X=X . B . + .      }
25 . IF . X <C . THEN . 21 . replaces the NEXT
```

Table 6 gives the mathematical expressions for LET statements. With the computer in the READY state, type

MOV . 0BB5 0C00 0C00

and then change to the high level language by pressing RS on the keyboard. The word READY will then be replaced by BURP. The RS key types in RUN . 0800, and initiates the high level system. The low level MOV command moves the sample program in r o m out into the r/w m., where it can be examined by typing

LIST . 5 .

which gives

```
005 FOR A=1 STEP 1 UNTIL 25—
006 LET L=A LOG—
007 PRINT A0 L8 —
008 NEXT A—
009 END—
OC4A
```

The dash shows where a line ends and virtually every term, including the last on each line, is followed by a space. The address 0C4A gives the upper limit of the program storage currently in use, and from 0C00 up to 1DC0 is available. Now type

RUN . 5 .

The computer should print the common logarithms of the numbers 1 to 25. When it has finished, the computer is ready for a command, indicated whenever BURP is the only word on the top line. Type

DEL . 6 .

and the program will list out with line number 6 deleted. Note that the end address is now 0C3B, i e. when lines are deleted, the computer reworks the remaining lines back towards the start of the memory space This makes best use of the memory and stops the build up of rubbish within the memory which would slow down the program execution Next,

ADD .
6 . LET . L=A . ROOT .

After typing the colon the word ADD will disappear, i e you are back in command The colon is necessary at the end of an ADD or a LOAD because it inserts the hex byte C0 at the end of the program block. This code tells the computer where to stop and go back from, when it is scanning through the memory. Now,

RUN . 5 .

which will list out the square roots of the numbers 1 to 25 Then,

DEL . 6 .
ADD .
6 . LET . L=A . EX .
RUN . 5 .

This program lists the natural anti-logs, e^x, of the numbers 1 to 25, and will show how the display switches over to scientific notation, the last result being 7.2004907×10^{10}. Although mathematically correct, these are rather crude presentations of the results. Type

ADD .
4 . PRINT . " . X . . . . . EXP X":
RUN . 4 .

which adds a heading above each of the columns, or,

```
DEL ␣ 7 ␣
ADD ␣
7 ␣ PRINT ␣ "THE NATURAL ␣ ANTI-
LOG ␣OF" A0, ␣" ␣IS"L4 ␣
RUN ␣ 5 ␣
```

which gives a different display format, see Table 7. Note that the comma after A0 suppressed the characters after the decimal point, rather than leaving a large gap. L4 means L, printed to 4 decimal figures, although without the compaction of the scientific results that a comma would bring. Try

```
DEL ␣ 7 ␣
ADD ␣
7 ␣ PRINT ␣ "THE NATURAL ANTI-
LOG ␣OF" A0 ␣" ␣IS"L4, ␣
RUN ␣ 5 ␣
```

to see the difference. If you make a mistake in these exercises, just terminate the line with a RETURN and type it in again. It is important, as already explained, that the LOAD and ADD commands are only left with a colon, do not be tempted to do so with an RS. If you have corrected a line in this way, when back in the command state, delete that line, and the computer will erase the first line it comes to with that number and then re-list with the second version in the correct place. Naturally, if you have mis-typed a line twice or more, this deleting procedure must be repeated until the correct line appears in place. The running of a program may be halted at any time by pressing any key on the keyboard, but as this returns the computer to the low-level, READY state, follow it with an RS for BURP.

Table 6. Mathematical expressions for LET statements.

Expression	Effect
+ − * /	Y+X→X, Y−X→X, Y×X→X, Y/X→X. In these four operations the stack collapses thus: Z→Y, T→Z, 0→T
YX	Y to the power of X→X. Stack collapses as above
REC	1/X→X, i.e., reciprocal of X. In this, and the following, Y, Z and T remain unchanged
ROOT	$\sqrt{X}$→X
SQ	X^2→X
TENX	10^X→X, i.e., common anti-logarithm
EX	e^X→X, i.e., natural anti-logarithm
LN	ln(X)→X, i.e., natural logarithm of X
LOG	log(X)→X, i.e., common logarithm of X
SIN	sine (X)→X. All trig. functions operate in degrees
COS	cosine (X)→X
TAN	tangent (X)→X
SIN−	$\sin^{-1}$(X)→X
COS−	$\cos^{-1}$(X)→X
TAN−	$\tan^{-1}$(X)→X
DTR	Converts X in degrees to radians
RTD	Converts X in radians to degrees
NEG	−X→X, i.e., change sign
PI	3.1415927→X
ENT	X→Y, Y→Z, Z→T. T is lost. X remains in X.
ROLL	Y→X, Z→Y, T→Z, X→T. Nothing is lost.
XEY	X exchanges with Y

In use, all of these expressions are followed by a space, e.g. for

$$X=\frac{1}{2\pi\sqrt{LC}}$$

```
LET ␣ X=L ␣ C ␣ * ␣ ROOT ␣ 2 ␣ * ␣ PI ␣ * ␣ REC ␣
```

Errors will occur in calculations under certain conditions:
LN or LOG when X is less than or equal to zero.
TAN when X is an odd multiple of 90° (90°, 270°, 450° etc.)
SIN, COS or TAN when |X| is greater or equal to 9000°
SIN− or COS− when |X| is greater than 1 or less than 10^{-50}
ROOT when X is negative
/ or REC when X = 0
or for any result less than 10^{-99} or greater than 9.9999999×10^{99}

Table 7. Print out giving natural anti-loge of numbers 1 to 25. The display switches to scientific notation at number 19.

```
THE NATURAL ANTI-LOG OF 1. IS 2.7183
THE NATURAL ANTI-LOG OF 2. IS 7.3891
THE NATURAL ANTI-LOG OF 3. IS 20.0855
THE NATURAL ANTI-LOG OF 4. IS 54.5982
THE NATURAL ANTI-LOG OF 5. IS 148.413
THE NATURAL ANTI-LOG OF 6. IS 403.4268
THE NATURAL ANTI-LOG OF 7. IS 1096.6332
THE NATURAL ANTI-LOG OF 8. IS 2980.9580
THE NATURAL ANTI-LOG OF 9. IS 8103.0839
THE NATURAL ANTI-LOG OF 10. IS 22026.466
THE NATURAL ANTI-LOG OF 11. IS 59874.143
THE NATURAL ANTI-LOG OF 12. IS 162754.79
THE NATURAL ANTI-LOG OF 13. IS 442413.40
THE NATURAL ANTI-LOG OF 14. IS 1202604.3
THE NATURAL ANTI-LOG OF 15. IS 3269017.4
THE NATURAL ANTI-LOG OF 16. IS 8886110.7
THE NATURAL ANTI-LOG OF 17. IS 24154953.
THE NATURAL ANTI-LOG OF 18. IS 65659970.
THE NATURAL ANTI-LOG OF 19. IS 1.7848    08
THE NATURAL ANTI-LOG OF 20. IS 4.8517    08
THE NATURAL ANTI-LOG OF 21. IS 1.3188    09
THE NATURAL ANTI-LOG OF 22. IS 3.5849    09
THE NATURAL ANTI-LOG OF 23. IS 9.7448    09
THE NATURAL ANTI-LOG OF 24. IS 2.6489    10
THE NATURAL ANTI-LOG OF 25. IS 7.2005    10
```

Loading programs

Programs are loaded by typing

```
LOAD ␣
```

and then the lines of the program, each of which must start with the number of that line. These lines do not have to be entered in the correct order, nor do all three digits of the number need to be typed in as they appear in the list. For internal reasons of the computer, it is not possible to have lines 0, 192, or 237. It is recommended that, for speed of execution, the lines used are kept fairly close together numerically, as this saves the computer scanning for lines which do not exist. In program development, it helps to initially use every third line number so that there is plenty of room for later additions. Remember that LOAD starts loading at the beginning of the program storage area and will thus erase any previously stored programs. If you want to add to the present lines, use ADD.

Entering data

When the computer comes across the program line INPUT, it goes to the next clear line on the v.d.u. and waits for you to enter the number of variables specified in the program line. Numbers entered must be followed by a space, except in the case of scientifically expressed numbers, which, because of the fixed length of the exponent, are recognised as terminated when the second exponent digit has been typed in. The l.e.d. associated with the keyboard is useful because it indicates whether the computer is, or is not, waiting for you to do something.

Finally, remember the spaces required during loading, and those after the three factors you type in during program execution.

To be continued

A scientific computer — 4

More programming in high and low level languages

by J. H. Adams, M.Sc.

THE MORTGAGE PROGRAM in Table 8 computes, from a given principal, annual interest rate and period for which a loan is to run (represented by P, I and T in the program), the monthly repayment and repayment schedule for a standard mortgage. The format closely follows that of standard BASIC. In line 6, an interest factor

$$K=1+\frac{I}{100}$$

is calculated, whilst the expression evaluated in line 7 is

$$B=\frac{K^T}{K^T-1}\times\frac{IP}{1200}$$

using the stack operation ENT to push K^T into the Y and Z registers of the stack as shown in Table 9. A special

Table 9 Stack operations for the mortgage program.

COMMAND	X	Y	Z	T
YX	K^T	-	-	-
ENT	K^T	K^T	-	-
ENT	K^T	K^T	K^T	-
1	1	K^T	K^T	-
-	K^T-1	K^T	-	0
/	$\frac{K^T}{K^T-1}$	-	0	0

print format is used in lines 13 and 19 to round the displayed values of B and P to the nearest penny.

Table 10 shows two separate programs cascaded into the programming area. The first is run by the command RUN 4 and is a game which simulates the landing of a rocket on Earth. Lines 4 to 8 set a fuel level of 120 (F), a velocity of −50m/s (V) and an initial height of 250m(H). After presenting this information, the computer waits for the player to type in a one second burn of fuel, B, which is checked against the present amount of fuel (line 14) and then used to reduce the velocity by B−5, provided that there is enough fuel available.

The aim, of course, is to simultaneously reduce the velocity and height to zero, without running out of fuel. The

Table 8 Print out of a mortgage program based on the high level language.

```
003 PRINT "                    ***MORTGAGE PROGRAM***"
004 PRINT "    INPUT PRINCIPAL, INTEREST RATE & TERM"
005 INPUT P I T
006 LET K=1 100 / 1 +
007 LET B=K T YX ENT ENT 1 - / I * 1200 / P *
013 PRINT "  MONTHLY REPAYMENT ="B2
014 LET B=P 12 *
015 PRINT "       ***REPAYMENT SCHEDULE***"
016 FOR X=1 STEP 1 UNTIL T
018 LET F=P K * B -
019 PRINT "AFTER"X0, " YEARS YOU OWE"P2, " POU  "
021 NEXT X
024 END

 ODED
```

Table 10 Cascaded programs for a rocket landing game and the solutions of F (x)=0.

```
.04 LET C=1
005 LET F=120
006 LET V=-50
007 LET H=250
008 PRINT "HEIGHT="H2, "  VELOCITY="V2, "M/S  FUEL LEFT="F0, "UNITS"
010 LET C=C 1 +
012 INPUT B
011 IF C<15 THEN 14
012 ERASE
013 LET C=0
014 IF B>F THEN 25
015 LET F=F B -
016 LET B=B 5 -
017 LET J=H
018 LET N=H V + B 2 / +
019 IF H<0 THEN 30
020 LET V=V B +
021 IF H=0 THEN 37
024 GOTO 8
025 PRINT "OUT OF FUEL  PREPARE TO CRASH."
026 END
030 LET V=V SQ J 10 * + RT
031 PRINT "YOU HAVE CRASHED AT"V2, "M/S."
032 END
035 PRINT "WELL DONE, YOU HAVE LANDED."
036 END
037 IF V=0 THEN 35
038 PRINT "YOU HAVE LANDED TOO FAST.  HAVE A NICE STAT "
039 END
098 PRINT "  THIS PROGRAM USES NEWTONS METHOD FOR SOLVING"
099 PRINT "THE FORMULA F = F(X).  ENTER AN INITIAL GUESS FOR X NOW. . ."
100 INPUT Q
101 ERASE
102 LET X=Q
105 GOSUB 200
110 LET Q=F
115 LET X=X 1.00001 *
120 GOSUB 200
125 LET T=Q SQ RT
127 TOP
130 IF T<0.000001 THEN 190
135 LET Q=1 F Q / 1 - REC 0.00001 * - Q *
137 PRINT Q8
140 GOTO 102
190 PRINT "THE SOLUTION IS X="Q6
195 END
200 LET F=X LN X 3 * + 10.6074 -
205 RETURN

100F
```

exercise is based upon standard Newtonian equations of motion; $s=u+\frac{1}{2}a$ and $v=u+a$. Crash velocities are worked out (line 30), using $v^2=u^2+2as$. In the program execution, C acts as a go counter, clearing the screen every 15 burns. This might seem unnecessary, as it takes some unusual playing to avoid a crash and not win in that number of attempts. There is a simple technique for predicting solutions to this game, but I will leave the reader to deduce this.

One of the most economical solutions uses burns of 0, 0, 0, 25 and 50. For a more daunting version, the 2 in line 18 can be made an inputted variable (which will affect the acceleration due to gravity) or, even more difficult, a function of the value of H.

The second program uses Newton's method to solve the equation $F(x)=0$. The equation in this case, $\mathrm{Ln}(X)+3X-10.8074=0$, is written at line 200 and, as it is required twice in the program, it is called as a subroutine at lines 105 and 120. Given an initial guess Q, at line 100, the computer calculates the next guess at Q by

$$Q-\frac{F(Q)}{F'(Q)}$$

calculated by the approximation

$$Q1-\frac{0.00001F(Q)}{F(1.0001Q)-F(Q)}$$

Line 125 assigns the absolute value of G to T, G being the difference between two successive values for Q and, if T is below the criterion of accuracy set in line 130, the program branches to line 190 and prints out a final rounded solution for X.

Note that if these two programs, or any material with more than 31 lines, are loaded, a LIST or DEL command will list the first 31 and then display LIST INCOMPLETE, preceded by the next valid line number on the top line of the screen. To display the rest of the program, or the next 31 lines, press the space bar.

Table 11 Program for analysing the pre-amplifier in Fig. 19.

```
005 INPUT A D B E C F
006 PRINT "FREQ          GAIN (DB)"
008 LET A=A 1000 *
011 LET B=B 1000 *
014 LET C=C 1000 *
017 LET D=D 1E12 /
020 LET E=E 1E12 /
023 LET F=F 1E06 /
026 LET W=PI 2 *
032 LET G=W D A * * SQ 1 +
036 LET H=W E B * * SQ 1 +
042 LET I=A SQ D * G / B SQ E * H / +
048 LET J=A G / B H / +
054 LET K=W C F * *
060 LET L=J I K * * - SQ W I * F J * + SQ + RT C / K K REC + /
066 LET L=L 60 / LOG 20 *
072 LET N=W 2 / PI /
078 IF N>20000 THEN 200
084 PRINT N1 L0
092 LET W=W 10 RT RT
053 GO 26
200 END

QE15 ,
```

Table 12 Computer run of results for the pre-amplifier.

FREQ	GAIN (DB)
1.0	6.
1.8	11.
3.2	15.
5.6	19.
10.	21.
17.8	21.
31.6	19.
56.2	16.
100.	12.
177.8	8.
316.2	4.
562.3	2.
1000.	0.
1778.3	-2.
3162.3	-5.
5623.4	-9.
10000.	-13.
17782.8	-16.

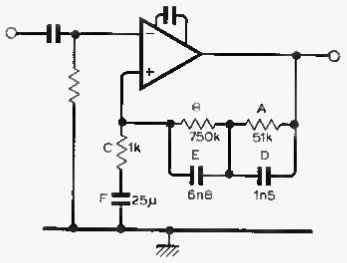


Fig. 19. *Typical RIAA equalised pre-amplifier based on the MC1303. The results of a computer run on this circuit are shown in Table 12.*

Table 13 Program for computing the intercept and gradient of the best fitting straight line.

```
001 LET A=0
002 LET B=0
003 LET C=0
004 LET D=0
005 LET N=0
006 LET E=0
007 INPUT X Y
008 LET N=N 1 +
009 LET A=A X +
011 LET B=B Y +
013 LET C=C X Y * +
015 LET D=D X SQ +
016 LET E=E Y SQ +
018 LET P=C A B * N / - D A SQ N / - /
019 LET L=B N / M A * N / -
021 PRINT 'AFTER'N0, 'PAIRS, M='M7 'C='L7
023 LET R=C A B * N / - SQ D A SQ N / - / E B SQ N / - /
024 LET R=R 100 *
025 PRINT 'COEFFICIENT OF DETERMINATION='R2, '%   '
027 TOP
029 GO 7

QDA7
```

Scientific numbers

The computer switches to a scientific display on numbers greater than 99,999,999 or less than 0.0001. Numbers appearing in programs or being entered in response to an INPUT line, may be entered scientifically or in floating point, provided that they are within the computers range. When entering scientifically expressed numbers, a space is not required at the end of the figures because the E entered in the figures tells the computer that only two more digits are to be entered. The standard form of one figure in front of the decimal point will always occur in displayed results, but need not be adhered to when entering because the computer recognises 1.00E02, 100E00, 0.01E04, .001E05 or 1000000E-04 as all being 100. This is demonstrated in the next program. Fig. 19 shows a recommended circuit for the Motorola MC1303 dual amplifier used as a RIAA equalised phono pre-amplifier. Tables 11 and 12 show the program for, and a run of, an analysis of the circuit. Values are entered in the most convenient units, resistors in kilohms, D and E in picofarads, and F in microfarads, and then scaled to their basic units in lines 8 to 23. The equations for working out the gain at various frequencies are,

$$G=1+(WDA)^2$$

$$H=1+(WEB)^2$$

$$I=\frac{A^2D}{G}+\frac{B^2E}{H}$$

$$J=\frac{A}{G}+\frac{B}{H}$$

$$K=WCF$$

$$L=\frac{((J-WKI)^2+(JK+WI)^2)^{1/2}}{C(K+1/K)}$$

The last equation is a good argument for Reverse Polish. Note that in line 26 π can be called as PI.

line 38 full use is made of the 4 level stack for storing intermediate values and results. This line actually consists of more than one v.d.u. line's worth of characters, and it therefore overruns into the next line

line 38 & 52 RT is an abbreviation of ROOT. In word recognition, the computer only considers the first and last letters of a word, which allows for considerable laxity in typing

When establishing the relationship between two sets of data, the first test is usually one of proportionality, i.e. will the data, if plotted, give a straight line? Table 13 lists a program which uses linear regression to compute the intercept and gradient of the best fitting straight line for a series of pairs of co-ordinates (horizontal, then vertical), read in at line 7. Each set of data updates the values of M and C, and also takes part in the calculation of a coefficient-of-determination, which gives a measure of the fit of the line to the co-ordinates. Note the use of the command TOP at line 27, which clears and resets the data entry point to the top of the screen each time.

Low level programming

When low level programming is used, charts of the type shown in Table 14 are very helpful for translating between the mnemonics for the Z80 operations and the actual hexadecimal codes. If the charts are used in conjunction with the technical manual for the MK3880/Z80, program assembly and disassembly is

Table 14 Conversion charts for the Z80 instruction set.

Second character of Z80 code

First character	0	1	2	3	4	5	6	7
0	NOP	LD BC nn	LD(BC) A	INC BC	INC B	DEC B	LD B n	RLC A
1	DJNZ	LD DE nn	LD(BC) A	INC DE	INC D	DEC D	LD D,n	RL A
2	JRHZ e	LD HL nn	LD(nn) HL	INC HL	INC H	DEC H	LD H n	DAA
3	JRNC e	LD SP nn	LD(nn) A	INC SP	INC(HL)	DEC(HL)	LD(HL) n	SCF
4	LD B B	LD B C	LD B D	LD B E	LD B H	LD B L	LD B,(HL)	LD B A
5	LD D B	LD D C	LD D D	LD D E	LD D H	LD D L	LD D,(HL)	LD D A
6	LD H B	LD H C	LD H,D	LD H E	LD H H	LD H,L	LD H (HL)	LD H A
7	LD(HL) B	LD(HL) C	LD(HL),D	LD(HL),E	LD(HL),H	LD(HL),L	HALT	LD(HL),A
8	ADD B	ADDC	ADD D	ADD E	ADD H	ADD L	ADD(HL)	ADD A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB(HL)	SUB A
A	AND B	AND C	AND D	AND E	AND H	AHD L	AND(HL)	AND A
B	OR B	OR C	OR D	OR E	OR H	OR L	OR(HL)	DR A
C	RET NZ	POP BC	JPNZ nn	JP nn	CNZ nn	PUSH BC	ADD n	RST D
D	RET NC	POP DE	JPHC nn	OUT A (N)	CNC nn	PUSH DE	SUB n	RST 16
E	RET PO	POP HL	JPPO nn	EX(SP) HL	CPO nn	PUSH HL	AND n	RST 32
F	RET P	POP AF	JPP nn	DI	CP nn	PUSH AF	OR n	RST 48

First character	8	9	A	B	C	D	E	F
0	EX AF AF	ADD HL,BC	LD A (BC)	DEC BC	INC C	DEC C	LD C n	RRC A
1	JR e	ADD HL,DE	LD A (DE)	DEC DE	INC E	DEC E	LD E n	RR A
2	JRZ e	ADD HL, HL	LD HL (nn)	DEC HL	INC L	DEC L	LD L n	CPL
3	JRC e	ADD HL,SP	LD A (nn)	DEC SP	INC A	DEC A	LD A n	CCF
4	LD C B	LD C C	LD C D	LD C E	LD C H	LD C L	LD C (HL)	LD C A
5	LD E B	LD E,C	LD E D	LD E E	LD E H	LD E L	LD E (HL)	LD E A
6	LD L B	LD L C	LD L D	LD L E	LD L H	LD L L	LD L,(HL)	LD L A
7	LD A B	LD A C	LD A D	LD A E	LD A H	LD A L	LD A,(HL)	LD A A
8	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC(HL)	ADC A
9	SBC B	SBC C	SBC D	SBC E	SBC H	SBC L	SBC(HL)	SBC A
A	XOR B	XOR C	XOR D	XOR E	XOR H	XOR L	XOR(HL)	XOR A
B	CP B	CP C	CP D	CP E	CP H	CP L	CP(HL)	CP A
C	RET Z	RET	JPZ nn	*	CZ nn	CALL nn	ADC n	RST 8
D	RET C	EXX	JPC nn	IN A (n)	CC nn	@	SBC n	RST 24
E	RET PE	JP (HL)	JPPE nn	EX DE HL	CPE nn	‡	XOR n	RST 40
F	RET N	LD SP HL	JPN nn	EI	CN nn	@	CP n	RST 56

@ DD or FD preceding underlined codes exchanges the operand IX or IY respectively for HL
In both cases the displacement implicit in an indexed operation follows the code
* ‡ CB and ED precede codes shown below

Op-codes preceded by CB

First \ Second	D	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	RLC B	RLC C	RLC D	RLC E	RLC H	RLC L	RLC(HL)	RLC A	RRC B	RRC C	RRC D	RRC E	RRC H	RRC L	RRC(HL)	RRC A
1	RL B	RL C	RL D	RL E	RL H	RL L	RL(HL)	RL A	RR B	RR C	RR D	RR E	RR H	RR L	RR(HL)	RR A
2	SLA B	SLA C	SLA D	SLA E	SLA H	SLA L	SLA(HL)	SLA A	SRA B	SRA C	SRA D	SRA E	SRA H	SRA L	SRA(HL)	SRA A
3									SRL B	SRL C	SRL D	SRL E	SRL H	SRL L	SRL(HL)	SRL A

Bit tests, 01xx xyyy (binary)
Reset bit, 10xx xyyy (binary) where xxx is the bit number yyy the register code
Set bit, 11xx xyyy (binary)

B—0 H—4
C—1 L—5
D—2 (HL)—6
E—3 A—7

Op-codes preceded by ED

	D	1	2	3	5	6	7	8	9	A	B	D	E	F
4	IN B (C)	OUT(C) B	SBC HL BC	LD BC (nn)	RET N	IM 0	LD I A	IN C (C)	OUT(C) C	ADC HL BC	LD(nn) BC	RET I		LD R A
5	IN D (C)	OUT(C) D	SBC HL DE	LD DE (nn)		IM 1	LD A I	IN E (C)	OUT(C) E	ADC HL DE	LD(nn) DE		IM2	LD A R
6	IN H (C)	OUT(C) H	SBC HL HL				RRD	IN L (C)	OUT(C) L	ADC HL HL				RLD
7			SBC HL SP	LD SP (nn)				IN A (C)	OUT(C) A	ADC HL SP	DL(nn) SP			

A	LDI	CPI	INI	OUTI			LDD	CPD	IND	OUTD
B	LDIR	CPIR	INIR	OUTIR			LDDR	CPDR	IHDR	OUTDR

To find the mnemonic corresponding to a particular hex op code look for the first digit of the byte down the side of the tables and for the second digit across the top
To find the op-code corresponding to a particular mnemonic reverse this process

quite easy. As an example, Table 15 shows an analysis of the first part of the BURP monitor starting at address 0800. There are many subroutines in the computer's operating system and these are useful when low level programs are being written. Table 16 lists the subroutines with their CALL addresses, mnemonics and a brief description of their functions.

Development and use of machine code programs is generally a matter of personal requirement and therefore the demonstration programs will probably be of little practical use to constructors. One, however, listed in Table 17, which might be of interest to other teachers, shows the results when quanta of energy are randomly swopped between 2048 atoms (as used in Nuffield A level physics). To generate the pseudo random numbers, a 17-bit shift register with its input being the exclusive OR of the 16th and 17th bits, is set up in the Z80. There are two versions, RUN 1CCD gives a display of the atomic matrix up-dated every 256 swops and RUN 1C00 does the same, but also totals, in decimal, the number of sites with one quantum, with two quanta etc. Modifying the byte 1C04 from 31 to 32 or 33 alters the initial filling up of the matrix from all ones to all twos or threes respectively.

Table 15 Operation of part of the BURP monitor.

Hex bytes	Mnemonic	Operation performed
31 DF 1F	LD SP 1FDF	Loads the Z80 stack pointer with 1FDF
11 00 80	LD DE 8000	Loads the 16 bit register pair DE with 8000, which is the address of the top left-hand corner of the v.d.u.
CD CE 03	CALL 03CE	Calls the subroutine at 03CE (see Table)
20 02 02 ... 1D		Data for the preceding subroutine (displays 'BURP')
E7	RST 32	A special one-byte CALL to a subroutine at 002D, its effect is to print a space
E7	RST 32	As above
D5	PUSH DE	Stores DE in a section of the r/w.m. using the stack pointer as a pointer to, and a reminder of, this storage stack
CD C4 03	CALL 03C4	Calls the subroutine at 03C4 (clears the rest of the v.d.u. top line)
D1	POP DE	Restores the stored value of DE to the DE register pair
3E 04	LD A 04	Loads register A with the byte 04
32 E01F	LD (1FE0),A	Loads memory location 1FE0 with the contents of register A
1E 08	LD E 08	Loads register E with 08. Screen address is set eight spaces in on the top line, ready for a command

Table 16

```
     *****SUBROUTINES IN MACHINE CODE*****

0254 LEAD  PROVIDES LEADER FOR TAPE.
0260 TCHAR RECORDS (A) ON TAPE AT 300 BAUD
027F PADD  (HL) + A SPACE TO TTY
028E AHEX  CONVERTS 4 BIT HEX TO 'ASCII'
029D ASCII 'ASCII' TO TTY CONVERTER
02A5 PHEX  INSERTS SHIFTS AND SENDS TO TTY
02CC       LOOK-UP TABLE FOR TTY
02EC PSPA  TYPES A SPACE
02F0 PNEW  CARRIAGE RETURN, LINE FEED + FIG SHIFT
0301 PCHAR UART FOR TTY
0317 LIST  LIST SUBROUTINE
0336 TIME  TIME DELAY, FOLLOWED BY LOP COUNT
0345 TCLR  CLEARS TOP LINE AND SETS TO 8000
034E 8SPA  ROUNDS SCREEN ADDRESS UP TO XXX0 OR XXX8
0355 INWRD INPUTS AND ENCODES KEYBOARD (LAST - FIRST)
0372 MSPA  SPACER USED IN LIST AND LOAD
0393 CLR   CLEARS THE SCREEN
039F DADD  (HL) + A SPACE TO VDU
03A9 DHEX  DISPLAY A HEX BYTE ON THE VDU
           (03A9 IF THE BYTE IS IN (HL), 03AA IF IN (A))
03C4 ELIN  CLEARS OFF REST OF CURRENT VDU LINE
03CE DLIST DISPLAY THE FOLLOWING DATA
03DE LADD  LOADS HL FROM KEYBOARD
03E7 INHEX READS IN AND FORMS HEX BYTE
03F6 IN    READS KEYBOARD AND CONVERTS TO 4 BIT HEX
```

Table 17 Demonstration machine code program which shows the results when quanta of energy are randomly swopped between 2048 atoms.

```
1C00 21 00 0C 36   31 23 7C FE   14 20 F8 D9   21 80 0C 11
1C10 80 80 7E 12   13 23 00 7C   FE 14 20 F6   11 00 80 3E
1C20 30 06 0E C5   F5 21 00 0C   01 00 00 F1   F5 BE 20 01
1C30 03 23 7C FE   14 20 F4 C5   E1 C5 D5 11   30 30 01 E8
1C40 03 CD 82 1C   53 1E 30 01   64 00 CD 82   1C 06 30 7D
1C50 FE 0A 38 05   04 D6 0A 18   F7 E1 72 23   73 23 70 23
1C60 C6 30 77 23   EB C1 CD 4E   03 F1 C1 3C   10 B5 D9 00
1C70 00 CD 97 1C   7E FE 30 28   F8 35 CD 97   1C 34 10 F1
1C80 18 89 A7 E5   ED 42 E1 D8   ED 42 1C 18   F5 10 F1 18
1C90 83 FF FF FF   FF FF FF C8   A9 7A E6 C0   EA A1 1C CB
1CA0 E9 CB 21 CB   13 CB 12 D5   E1 7C E6 07   C6 0C 67 C9
1CB0 21 00 0C 36   31 23 7C FE   1C 20 F8 06   00 CD 97 1C
1CC0 7E FE 30 28   F8 35 CD 97   1C 34 10 F1   D9 21 00 CC
1CD0 11 00 80 7E   12 13 23 00   7C FE 14 20   F6 D9 18 DB
1CE0 FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF
```

Tape interface

The tape commands operate in the low level language, therefore, if a high level language program is to be recorded, its final address must be noted from a high level LIST. When recording it is worth spacing the blocks of recorded data because a 2 kilobyte block only requires 45 seconds of tape, and individual blocks are then easier to find. The leader of stop bits recorded automatically at the start of each recording lasts for about four seconds, so, when a recording is to be read into the computer, cue the tape just into this leader, type READ XXX, i.e. the first three characters of the hex address, start the tape and then type the last digit of the address.

In the kit of parts available for this design, one of the panel l.e.ds monitors the data stream and is turned on by the stop bits to indicate by flickering that data is being read in and, by steady illumination, that the recording has finished.

The TAPE command leaves the recording tone in the stop state so that, after the four second trailer, when the computer returns to the READY state the tone is left in the correct state for the next recording. When this trailer is reached during a READ, the computer must be interrupted by pressing a key.

Although the receiver is fairly flexible about frequencies and gives a 1 or 0, depending upon which side of 2kHz the tone is, the input from the tape recorder should be at least 1V r.m.s. For recording, the output variable resistor should be set so that, without overloading the input of the tape recorder, it is possible to over-record by a few dB, quantity rather than quality being the main criterion. There is no fine adjustment of the generated frequencies because of the flexibility of the receiver design. Several different interfaces and tape decks have been tried, but a consistent error rate has been impossible to establish, even with a judiciously placed finger slowing down the tape transport.

To be continued

A scientific computer — 5

Graphics, graph plotting and e.p.r.o.m. programming

By J. H. Adams, M Sc

THE SET OF CHARACTERS chosen for the graphics option was selected to augment the standard ASCII set and also provide various shapes for constructing diagrams and pictorial displays, see Fig. 20. The characters are programmed into a 2708 e.p.r.o.m. so the set may be easily altered to suit individual constructor's requirements. To make the shapes continuous, a full 10 × 6 dot structure is used for each character cell. This is achieved by loading bit 6 of IC_{45}, which was wired to 0 and thus inserted a one dot gap in between adjacent characters, and by disabling the line from IC_{34} pin 11 to the video gate IC_{33b}. The function of the gate was to blank the ninth and tenth line scan in each row of displayed characters to provide line spacing between those rows.

The 2708 is connected in parallel with the 2513 ASCII generator while bit 6 of the v.d.u. r./w.m. is fed directly to pin 11 of IC_{47} and through an inverter to IC_{100} pin 20. This bit selects which one of the character generators is enabled as shown in Fig. 21. The pulse at IC_{45} pin 1, whose trailing edge loads the shift register IC_{45} with the character dot pattern, is inverted so that its leading edge clocks the r/w.m. bit 6 into latch IC_{102}. The latch output feeds the video gate and sets an input which, on lines 1 to 8 of the character scan, is held low by the signal from pin 11 of IC_{34} applied to the S input. On the ninth and tenth scan when the S input is high, the video gate input will reflect the D input to the latch and thus the nature of the character being displayed, i.e. ASCII or graphic. Note that the latch must be loaded ahead of IC_{45} otherwise time delays in the system will cause the bottom left-hand corner of any graphic following a non-graphic to be blanked. When IC_{100} is disabled, the inverter on its D_5 output provides a low to IC_{45} pin 6 which then retains the original character spacing on ASCII.

Using graphics

As mentioned in part 3, ASCII may be directly loaded in low level language by opening a [and then typing the characters required. To enter graphics from this mode, open another [and then type in the graphic characters required. After typing the first character, which replaces the [on the screen, a cursor appears in the next screen position (a cursor is essential for picture construction) and the / key becomes a "rub-out and backstep" key for correcting errors. Typing a] reverts the computer to the ASCII mode of loading. With the high level language the computer is already in the ASCII mode and so the first [mentioned above need not be typed in. When loading ASCII or graphics in the low level, the computer includes], 1D in hexadecimal, in the string of characters loaded into the program. This is recognised as the end-of-string marker by the low level subroutine at $03CE_H$, mentioned in part 4, and for this reason a] can not be included in an ASCII string. In graphics, the [key stands for a shape and so this restriction does not apply, also, RETURN does not function so for a new line type], RETURN, [. All of the instructions for graphics are already in the monitor r.o.ms so no re-programming is required for this optional facility. A selection of computer displays is shown in Fig. 22.

Graph plotting

In part 4, a program was described which analyses the frequency response of an RIAA equalised pre-amplifier. Fig. 22(d) shows a version of this program which displays a graph of gain and frequency. Both quantities are logged before they are plotted to produce the standard dB versus log. of frequency format. The frequency is stepped logarithmically at line 52 at four increases per decade, and for each point plotted the frequency is displayed at the top of the screen. The original reason for developing this program was to examine the low-frequency gain of my hi-fi and so, after reaching 20kHz, the program branches to line 200 where it inputs a new value for F, the 25μF capacitor, and then replots a curve super-imposed upon the original.

When the graphics option is installed, the demonstration programs at the third end of the third e.p.r.o.m. in the computer can be replaced with firmware which enables it to accept two extra high level commands,

AXIS a b

which declares the maximum values of the vertical and horizontal axes of a graph plot, and

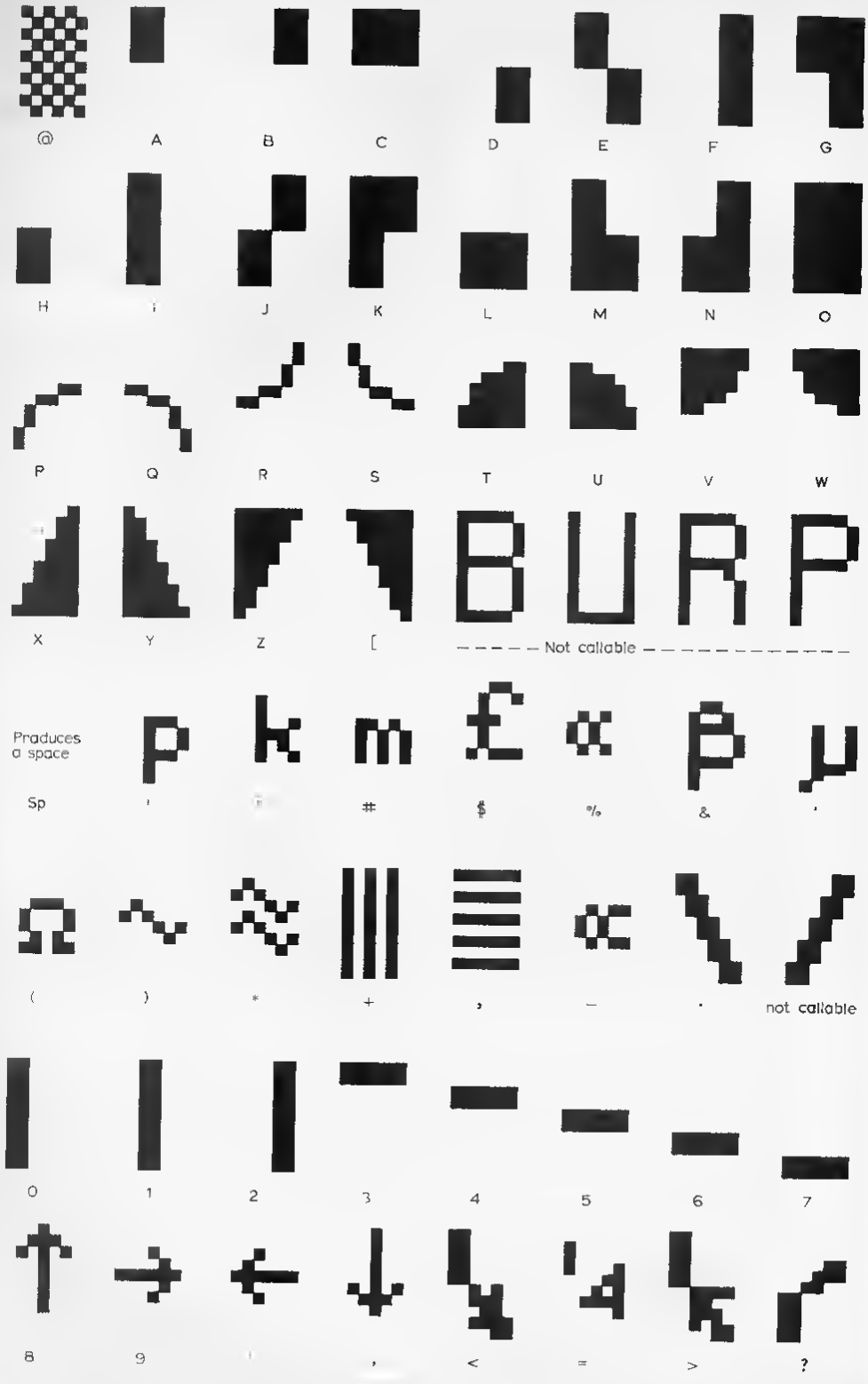
GRAPH a b

which plots the point (b,a) on a graph defined by the last AXIS statement. The terms a and b may be either variables or numbers. The plotted graph is of the first quadrant, i.e. both a and b positive, and is realised by dividing the screen into 8192 (128 by 64) cells which are selectively illuminated using graphic shapes at screen addresses obtained by scaling actual results to be plotted against the limits declared in the AXIS statement. The graph appears as discrete points, the spacing of which is usually determined by a FOR loop containing the GRAPH statement. The firmware controlling these processes ignores the sign of the variables (hence the first quadrant only) and will plot all results as positive. It also uses the variables P and Q for the scaling factors, thus only the remaining 24 are available in graph plotting programs.

E.p.r.o.m. programmer

The 2708 seems to be the most popular e.p.r.o.m. at present and this is probably due to the ease with which it may be programmed. Unlike many of its competitors, address locations and the data to be programmed at those locations are entered at the same level and polarity, and at the same pins as those obtained during read operations. Apart from the application of data to the device, the only changes made during programming are that the chip enable input is taken to +12V and a +25V pulse is applied to the programming pin 18.

It takes 100 ms to program each byte but, unfortunately, the device cannot be programmed byte by byte because pulses this long may cause spurious programming of adjacent cells on the chip. The specified maximum pulse length is only 1ms and so a programmer must make at least 100 "laps" of the device, slowly bringing up the values in each cell to their final state. The programmer requires 8 data bits, 10 address bits and a signal bit to commence the programming pulse for each byte. In Fig. 23, the two upper bits of the addresses are supplied by a four-way switch and the remaining 16 bits are fed serially to a 16-bit shift register formed by two 4015s. Data is sent out via the buffered D_7 line using OUT commands, and pulses to clock the bits into the shift


@ A B C D E F G
H I J K L M N O
P Q R S T U V W
X Y Z [B U R P
Not callable
Produces a space
Sp ! # $ % & '
( ) * + , - . not callable
0 1 2 3 4 5 6 7
8 9 : ; < = > ?

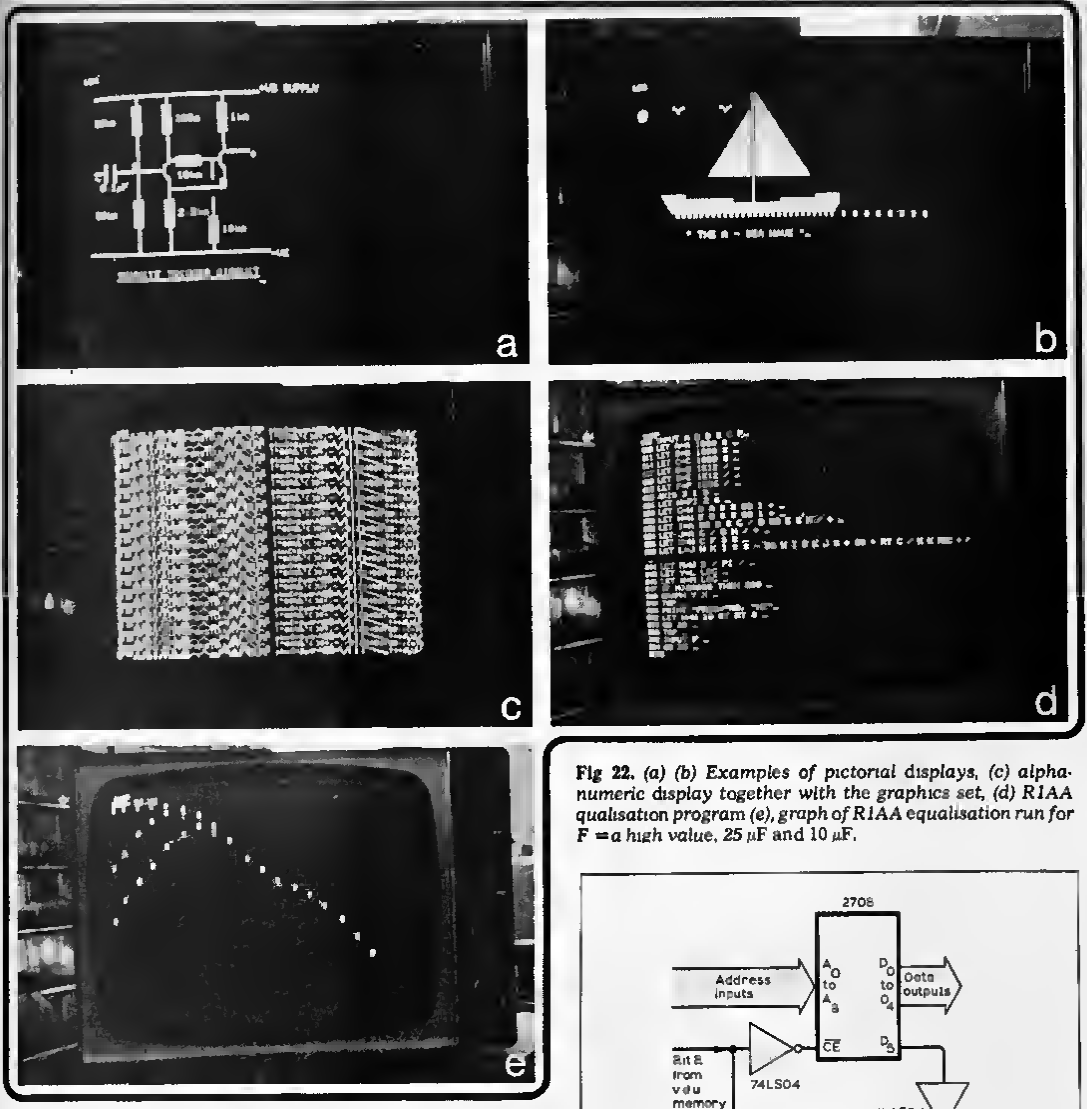
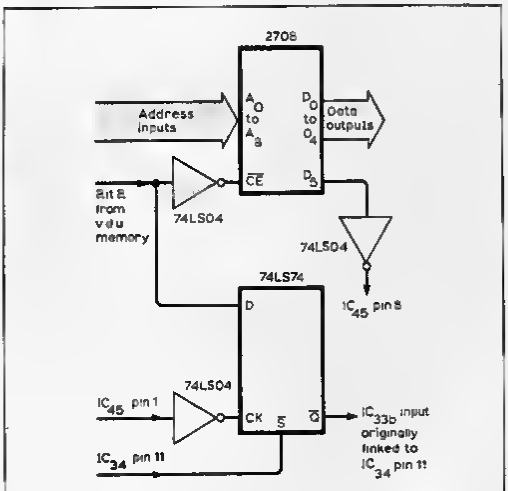


Fig 22. *(a) (b) Examples of pictorial displays, (c) alphanumeric display together with the graphics set, (d) RIAA qualisation program (e), graph of RIAA equalisation run for F = a high value, 25 μF and 10 μF.*

register are provided by the output write decoder IC_2. A second line from IC_2 triggers the 1ms monostable when the 16 bits are in place and initiates the programming. A hardware time delay leaves the Z80 free to prepare the next string of data for transmission to the shift register. This small reduction in time is important because during the programming over 100,000 operations are carried out, each with its own period of data transmission and 1ms pulse. Although it would save time to transmit the data in parallel, the extra connection complexity was considered unnecessary. For reliable programming some extra laps should be included to allow for monostable time delay variation. Unless the full 100ms per location

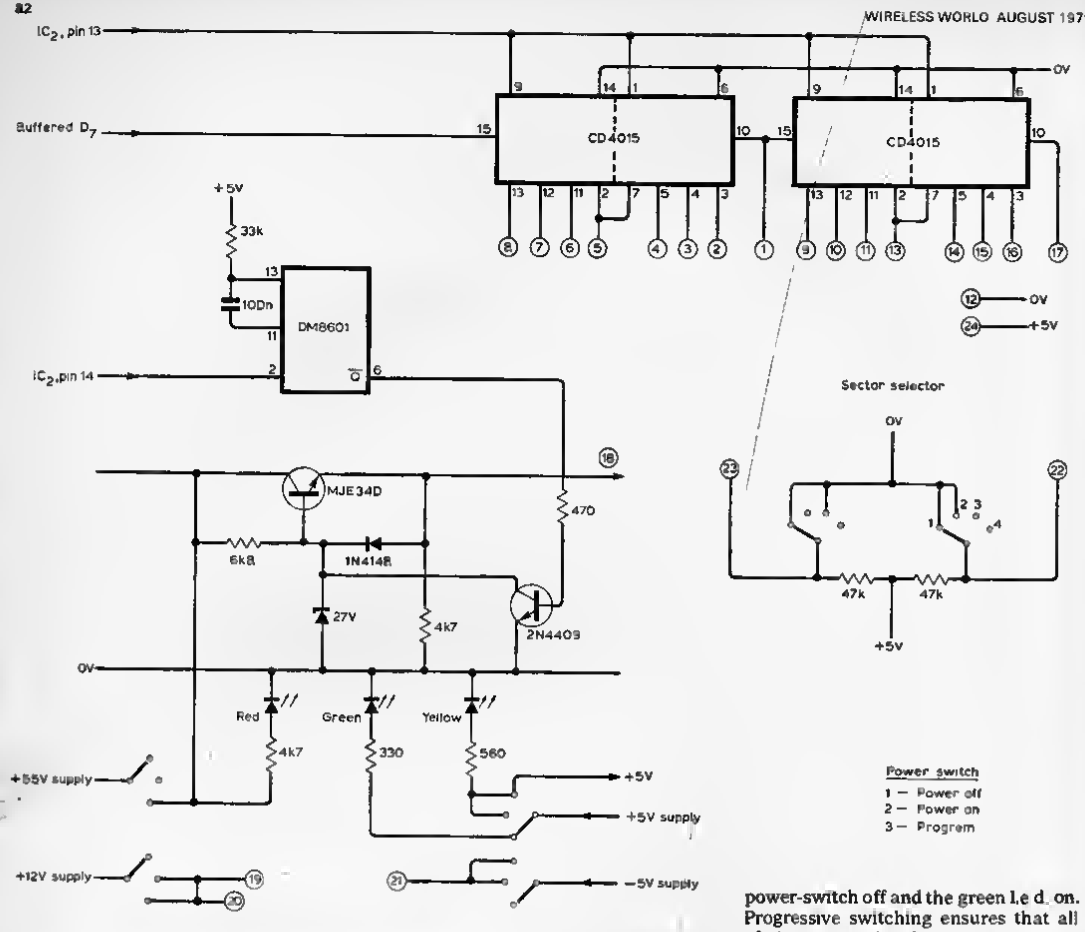
◀ **Fig. 20.** *Graphics character set.*

Fig. 21. *Optional graphics circuitry. All of the instructions for graphics are already in the monitor rom.* ▶



27D8 connections

pin	connects to	pin	connects to
1	IC_{47} pin 20	9	IC_{47} pin 4
2	" " 19	10	" " 5
3	" " 18	11	" 6
4	" 17	13	" 7
5	IC_{34} pin 11	14	" " 8
8	IC_{47} pin 16	18	0V
7	" " 15	22	IC_{47} pin 22
8	" " 14	23	" " 21



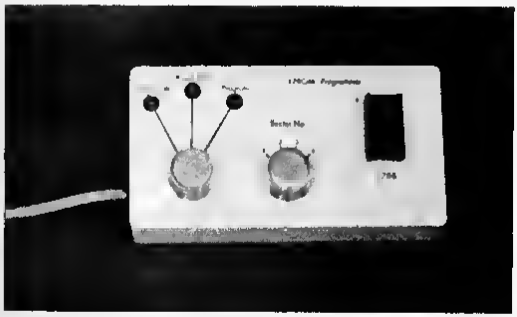


Fig. 23. *E.p.r.o.m programmer and circuitry Pin numbers for the i.c. socket are shown in circles.*

in applied some bytes may exhibit access times well outside the manufacturer's specification and this can cause erratic computer operation. The programming voltage is supplied through a conventional series-pass regulator and because at low levels pin 18 of the e.p.r.o.m. is a current source, the IN4148 and 2N4409 form a positive clamp.

Data to be programmed into an e.p.r.o.m. is assembled into the memory area 1C00 to 1CFF inclusive. Originally the memory was divided into four sectors and these were programmed separately which reduced the programmer address bits to 8, to match the shift registers. However, it has also proved useful when adding to or modifying small areas of r.o.ms during firmware development. When the r.o.ms are new or erased, all of the locations hold a 1, i.e. the bytes are all FF. If a sector of a 2708 is not to be completely programmed, or if a partially filled sector is to be added to, the command FILL 1C00 will fill 1C00 to 1CFF with FFs and will thus mask any areas which are not to be programmed prior to entering the data. The e.p.r.o.ms must only be inserted or removed from the programmer with the power-switch off and the green l.e.d. on. Progressive switching ensures that all of the required voltages are present before +25V is made available and prevents any chance of random programming. To program a sector, the command is PROM$_0$ but before this is used, check that the sector switch is in the required position. The programmer will also accept 2704 devices which are half the size of 2708s and only require sectors 1 and 2 although, in my experience, many 2704s are 2708s in another guise. All of the instructions for programming e.p.r.o.ms are in the original r.o.ms so no firmware changes are required for this option.

Erasing e.p.r.o.ms

For rapid erasing a high-power short wavelength ultra-violet lamp is required. Because commercial units are quite expensive I use a Philips 6 Watt u.v. lamp. The bulb should be coated in aluminium foil, except for a small window a third of the way from the glass end, to near the base. Because the base forms one of the connections, the foil must not touch it. This arrangement gives eye protection and concentrates the light at the window. The e.p.r.o.ms to be erased are placed as close as possible to the window and left for approximately three hours.

To be continued □

A Scientific Computer — 6

Final program examples, tv interface and radio teleprinter interface

By J. H. Adams, M.Sc

THE FINAL TWO programs in Table 18 were used to ease the design of active filters for the teleprinter interface. The filters are based around the LM3900 quad Norton amplifier i.c. and a low-pass version is shown in Fig. 24. The display in Fig. 25 shows a run of the first program which computes the resistor values required for a given gain, Q and fixed capacitor values. The program will intervene if the ratio of the capacitors is too low for correct operation, and when presenting the results it uses two graphic characters to enhance the appearance. New values for C_1 and C_2 may be entered repeatedly until the resistors are near enough to preferred values for the accuracy required.

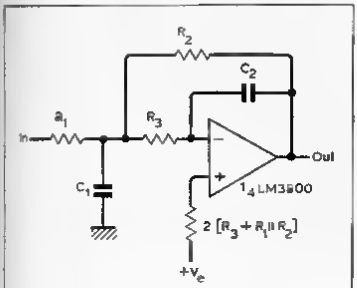


Fig. 24. *Low pass filter used for the teleprinter interface.*

```
002 PRINT "  LOW PASS ANALYSIS"
003 PRINT " INPUT R1, R2, C1, C2 NOW . . . "
005 INPUT R,S,C,D
006 AXIS 125,6
008 LET F=1
011 GOSUB 232
014 LET E=LOG G -
017 GOSUB 232  -
020 LET G=G E +
023 IF G<0 THEN 32
024 LET Z=F LOG
026 PRINT "F ="F1, "HZ"
027 GRAPH G,Z
028 LET F=F 10 RT *
030 GO 17
032 HALT
034 ERASE
036 GO 1
232 LET X=1E12 2 / PI / F / C /
235 LET Y=1E12 2 / PI / F / D /
238 LET A=X Y / R S * X / Y / - 1 - SQ
241 LET B=X R S + - Y / SQ    -
244 LET G=A B + ROOT REC LOG 20 *
247 RETURN

0000
```

```
004 PRINT "   LOW PASS FILTER PROGRAM"
005 PRINT " INPUT GAIN + Q REQUIRED, CORNER FREQUENCY, C1 + C2"
020 LET W=2 PI * F *
023 LET C=C 1E12 /
026 LET D=D 1E12 /
029 LET S=1 K Q / - ROOT 1 + 2 / Q / K / W / C / 1000 /
032 LET R=S A /
035 LET T=W C * SQ S * R * REC 1E00 /
038 PRINT "R1 ="R2, "kΩ,  R2 ="S, "kΩ,  R3 ="T, "kΩ"
041 INPUT C,D
044 GO 11
047 END
050 PRINT "REDUCE THE RATIO OF C2 TO C1.  INPUT NEW VALUES NOW . . ."
051 GO 36
053 INPUT D
056 LET D=D 1E12 /
059 GO 29

0000
```

Table 18. Two programs for designing active filters based on the LM3900.

Fig. 25. *V.d.u. showing a computer run for the circuit in Fig. 24*

The second program can then be used to give a gain versus frequency plot of the filter's characteristics. In Fig. 24, R_3 is assumed to be equal to R_1. The program terminates when the gain falls below unity, but it may be re-run by pressing the space bar as described in part 4.

A specific tv interface

It was mentioned in part 2 that if a live tv chassis is used for the v.d.u., a mains isolation transformer is necessary. This expense can be avoided by using a television set such as the Ferguson or Ultra model 3845 which has a fully isolated power supply. A simple interface for this set is shown in Fig. 26. All of the connections are made to the tube base except for the sync input. Resistor R_{72} on the tv's p.c.b. is lifted at the end nearest to the back of the set, and a wire from the empty pad is taken to a changeover switch which connects the original sync, or that from the computer

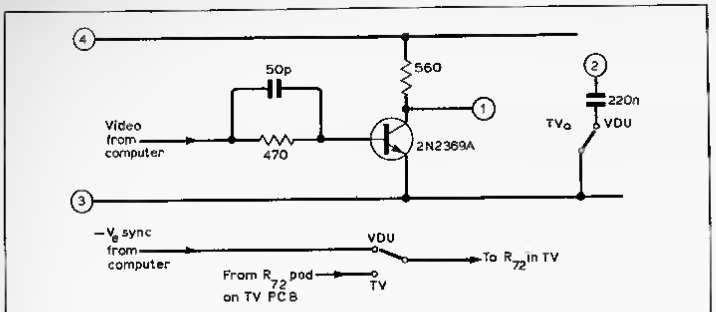


Fig. 26. *Tv interface for the Ultra model 3845. The numbers in circles are pin connections on the c.r.t. base.*

to the p.c.b. With negative going sync. supplied from the computer, a display should be obtained which, with the switch in the v.d.u. position, synchronises without any adjustment to either the horizontal or vertical holds. It is possible that the display will not completely fit onto the screen, but this can be rectified by the following alterations.

Loosen the polythene clamp which holds the scanning coils onto the neck of the tube and slide the plastic end of the width adjusting collar (this fits between the coils and the tube neck) to reduce the width of the picture. If the display is still too wide, connect a voltmeter between pins 3 and 4 on the tube base and adjust the preset resistor R_{69} which is just in front of the line timebase compartment. This sets the h.t., normally 11.3V, and should produce the required change with less than a one volt alteration. If the display is not horizontally centred, adjust the position of the core in the line oscillator coil (this can be found behind the line compartment nearer to the centre of the set than the "set h.t." preset). This must be done with a non-metallic tool.

If the television is tuned to a station whilst in use as a v.d.u., the tv picture may faintly modulate the display. This can be avoided by tuning off the station, unplugging the aerial or turning down the contrast control. If a brighter display is required, a higher supply voltage to the 2N2369A will be necessary. This can be supplied by the computer or from the 90V supply on pin 6 of the c.r.t. base. If the 90V is used, a potential divider must be connected because the maximum collector voltage of a 2N2369A is well below 90V. Whichever supply is chosen, the collector load should be increased proportionately from the 560Ω shown in this design.

Radio-teleprinter interface

One task for which this computer is well suited is the reception and transmission of radio teleprinter signals — RTTY. The system described here consists of hardware which converts f.s.k. (frequency shift keying) signals into digital levels, and software which is used to receive the serial stream of data, recognise the start of bytes and then frame, translate and display them. This software allows the Baud rate, the style of encoding and the code used to be controlled by the programmer.

The receiver should use a product detector or a b.f.o. so that the f.s.k. signals are available as two audio tones. The difference in frequency between the tones will depend upon the shift being used by the sender of the RTTY, but will usually range from 170 to 850 Hz. If the receiver's bandwidth can be reduced to this extent, the wanted/unwanted signal ratio will be improved and this will produce more reliable decoding. However, even with conventional bandwidths, the first two amplifiers in Fig. 27., which are connected as high and low-pass active filters with gains and Qs of one and a bandpass of approximately 1200 to 2100 Hz, will filter out the required tones to some extent. The filtered signal is then fed into a 565 phase locked loop. This i.c. compares the input signal with

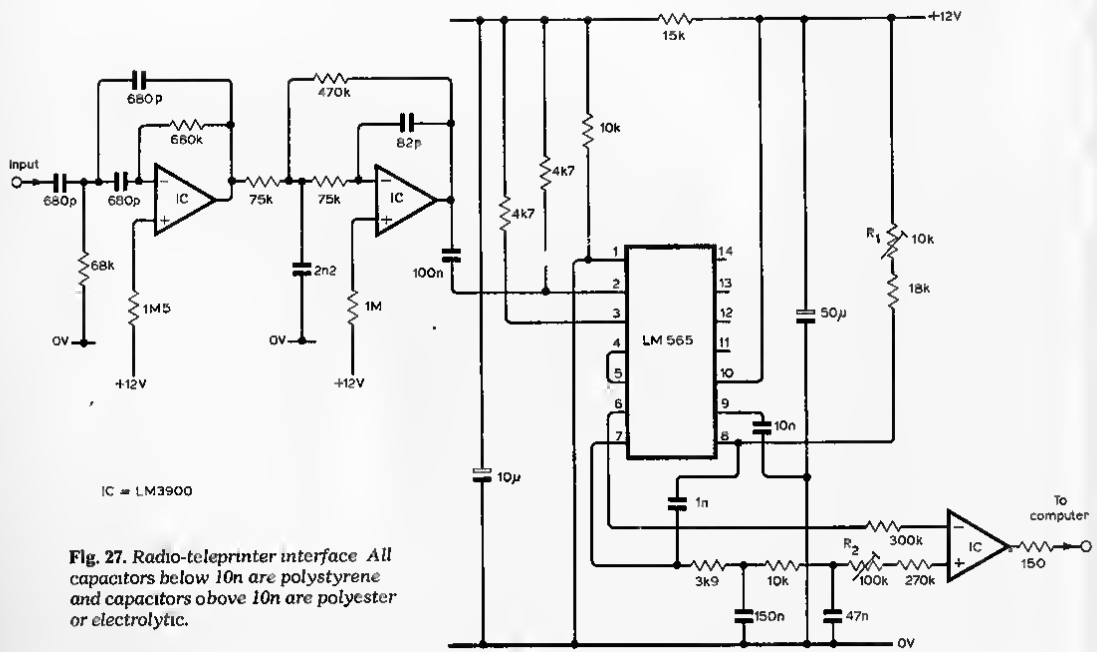


Fig. 27. *Radio-teleprinter interface. All capacitors below 10n are polystyrene and capacitors above 10n are polyester or electrolytic.*

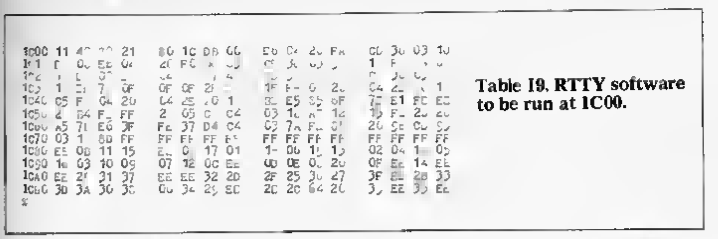


Table 19. RTTY software to be run at 1C00.

that of a v.c.o. which has been set at a frequency between the two tones. The result of this comparison will contain a low frequency component which after filtering is used to pull the oscillator to the frequency of the incoming tone. This voltage therefore switches in sympathy with the tones and is fed to a comparator which produces the digital output.

If reception only of RTTY is required, the oscillator's free-running frequency should be set at the midpoint of the active-filters pass band. Alternatively, when a complete terminal is used and standard frequencies are required, e.g. 1275 and 1445 Hz for a narrow shift, the oscillator is set to the midpoint of these frequencies. Before this adjustment is made, R_2 should be set so that with no input to the unit the output is on the point of changing state. With an input of 10mV at the required change-over frequency, R_1 should be set in the same way.

A 12V supply is available from the computer and the output of the unit should be fed, via the serial input buffer IC_{53}, to the line which feeds data bus line D_2.

A RTTY byte consists of a start bit, 5 data bits and 1½ stop bits. The software makes two checks 1ms apart for a start bit, to improve the noise immunity, and when it is successful in finding a start bit the data byte is read in, translated into ASCII and displayed. As there are two sets of characters in the RTTY, Murray code numbers and letters, the computer is set initially to letters and section 1C39 to 1C48 of the program, see Table 19, recognises and acts upon the Murray bytes to draw the correct ASCII byte for display. Because most teleprinters have more than the v.d.u's 64 characters per row, the section 1C5D to 1C72 ignores conventional carriage return and line feed bytes, but looks for spaces in the last eight character positions on a row. When one is found a new line is called which avoids the split words and short lines that occur with most v.d.u. systems. If this facility is not required, e.g. when receiving lines of tabulated data, alter 1C5F to 18, 1C9D and 1CBD to FF to obtain a new line with every carriage return received. Section 1C80 to 1CBF contains the look-up tables for converting letters (1C80 to 9F) and numbers (1CA0 to BF) from Murray to ASCII. Note that without the graphics option, the Murray £ appears as a $. Tuning the receiver is most easily done by observing the output of the unit on an oscilloscope and tuning for the cleanest display, although after some practice the ear is quite adept at picking out the correct point. If after correct tuning the display is random, this may be due to the transmission rate being other than 50 or 45.5 Baud. In this case the byte XX following the calling of the software time delay (CD 36 03 XX) must be altered to change the delay length, see part 4 for details of the sub-routine. The time delay has been set at 21ms as a compromise between 45.5 and 50 Baud by calling two delays with an XX of A0 between data bits, and three delays of 9B at the beginning to clear the start bit when recognised, and to return half way into the first data bit. Three times 9B plus the 10 time delay called between the two start bit checks equals 1½ times two A0 delays. If the code makes no sense when receiving at the correct Baud rate, it is probable that the sense of shift is reversed which causes the computer to read 1s as 0s and vice-versa. This can be corrected by changing sidebands on a receiver using a product detector, or by tuning the b.f.o. to the tones on the other side of zero beat. This problem is most easily recognised by characters appearing regularly but without spaces which suggests that the timing is correct, or if the test signals often transmitted by such stations are received as SYSYSYSY ... instead of RYRYRYRY. Narrow shift RTTY transmissions can usually be found between 14080 and 14100kHz during most of the day. □

Points arising

Unfortunately the following errors occurred in part 1 of the series. In the memory circuit of Fig.1, D_0-D_8 should read D_0-D_7. The 1k2 and 220Ω resistors on the base of Tr_1 in Fig. 2 should be transposed. For the kit, move R_2 to R_1 and fit a 220Ω in the R_2 position. Pin 15 of IC_{45} should be connected to 0V and to maintain the correct order for D_0-D_7, the two outputs from IC_{45} to IC_7 should be transposed. In the diagram on the left of Fig. 3, the NO arrow should be placed above the Read strobe box.

Two points have arisen from comments by constructors of the computer. Clock instability causing horizontal jitter on the picture can be eliminated by changing the 470Ω resistors at IC_{28} to 1kΩ. Due to manufacturing tolerances in some 4528 monostables, the tape reader may not function.

If this occurs replace the 27kΩ resistor connected to the 4528 with a 50kΩ preset, feed a 1V r.m.s. sine wave at 1800Hz into the tape reader and adjust the preset until the 4013 output which connects to the monostable is just changing state. The preset can then be replaced with the nearest preferred value resistor.

In conclusion

This section completes the scientific computer series. Due to a lack of space the firmware details have not been published but readers who wish to program their own r.o.ms can obtain a firmware list by sending a s.a.e. to this office.

We hope to support this design in the future with a floppy-disk drive and further practical programs. However, readers are invited to submit details of their own modifications or programs for publication.

We understand that John Adams is prepared to undertake the service or repair of computers built from a kit. Constructors experiencing difficulties should contact the author at 5 The Close, Radlett, Hertfordshire (Radlett 5723).

SIXTY YEARS AGO

After World War 1, amateur radio enthusiasts were left wondering for a time what was to happen to them. Then, as reported in the September 1919 issue, all pre-war licences were cancelled, preparatory to sweeping changes being introduced. In the present day uncertainty about citizen's band, one hears remarks which could have applied to the situation of sixty years ago.

"... said an official of the General Post Office to a Daily Express representative, we are very careful to see that applicants are not simply out for amusement. Before the war we had a good deal of trouble with silly fools who apparently asked their best girls to tea and amused them by sending out 'S.O.S.' signals on their wireless sets. Now that there have been such improvements in the apparatus inanities of this kind will be very obstructive to messages of commercial importance."

All we have to say with regard to that statement is that any practical wireless man knows how the "fools" could be soon discovered and eliminated, no ban has been placed upon private motor cars simply because a "fool" occasionally runs over a pedestrian. As for amusement, we should like to know why people may not amuse themselves in any way they please, provided they do not infringe the liberties of other citizens or transgress against the law. The amateur is willing to be supervised and to conform to reasonable regulations, he is even willing to act as a policeman of the aether amongst his own class, but judging from the look of things there exists a desire to obliterate him altogether. America has bowed — stiffly, it is true — to Prohibition, but she did not agree to the proposal to stifle amateur wireless telegraphy. Why should you? What are the Wireless Clubs doing to defend their rights? It was the American amateurs themselves who won their day, though they interested their legislators in the matter. Have our Wireless Clubs knowledge of any Member of Parliament who will stand up for them in the House?

More on the scientific computer

Further details of the monitors

By J. H. Adams, M Sc

After publication of the scientific computer series (April to September 1979) there have been many requests for more information on the firmware. This article describes in more detail the machine code and BURP monitors in terms of hexadecimal machine code. Readers will need a hex print-out of the three p.r.o.m.s and the mnemonic to hex conversion tables published in the July 1979 issue of *Wireless World*.

Several readers have expressed incredulity at the thought of working directly in machine code rather than using assembly language mnemonics. However, the hex codes for 50 to 60 of the most regularly used operations can soon be learnt and, thanks to the logical distribution of codes to operations, many more follow from these. The once-in-a-megabyte ones such as IN D (C), ED 50 in hex, can be obtained from the conversion table. This does not rule out working in assembly language and using an assembler, or translating yourself, but in my experience the latter soon becomes tiresome and it is easier to write in hexadecimal.

When writing software it is useful to have a supply of the forms shown in Fig 1. The instruction 18 a relative jump, should be pronounced one eight and not eighteen. Similarly, the second byte is one seven and definitely not seventeen. If you want to jump forward with a relative jump, simply make the jump byte the number of bytes (up to 7F) over which execution must move, in this case 17 — 1 row and 7 bytes, to reach the target byte FF. For a jump back to the same target from the second 18, calculate the jump forward code to the next byte immediately under the target, 02 in this case, and then jump up row by row, decrementing the higher order hex character, i.e. from 02, F2, E2. When using a jump back the byte must be in the range 80 to FD (FE and FF serve no useful purpose).

Fig 1. Typical software form

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
					18	17									
														FF	
										18	E2				

Work No

Machine code monitor

Both monitors follow the same basic sequence as illustrated in Fig. 2. With the machine code monitor the base address of the Z80 stack is set, the address for the top corner of the screen is loaded into the DE register pair which is then used throughout the monitor as the destination pointer or vector for v d u operations, and the message READY is printed by the subroutine at 03CE. This is one of several routines in the computer which draws data from the locations directly following the call of the routine. The program counter, which will have been pushed onto the stack is exchanged with the contents of the HL register pair and then used as a pointer to that data before being exchanged back onto the stack, at the end of the routine, to cause a return to, in this case, 0010. The start procedure then clears the rest of the top line, resets the teleprinter output flip-flop and, using the subroutine at 0355, reads in and encodes a command from the keyboard. As explained in table 1, only the first and last letters are important to the subroutine. Whilst this limits the number of possible combinations which will produce different codes, a byte by byte comparison with a look-up table comprising all of the commands would use far too much p r o m space. After this has been achieved (001A), a comparison is made and if the code is not FC (the entry code for RUN) executions jump over 0D bytes for a further comparison and so on until a match is found, whereupon a block of instructions is executed before operation reverts to 0000 again.

Table 1. Low level monitor subroutines

Address	
0254	Sets tape interface tone to 2400Hz and then calls 255 long time delays — about 4 seconds
0260	Transmits the byte in register A to the tape interface preceded by a start bit and followed by two stop bits
027F	Calls a new line and then prints the contents of HL on the teleprinter
028E	Formats the hex byte in register A for printing as two characters on the teleprinter
02EC	Prints a space on the teleprinter
02F0	Calls a new line on the teleprinter
0301	Prints the contents of the A register on the teleprinter
0317	List subroutine. Entered at 0317 the starting address must be loaded in from the keyboard. Entry at 0310 assumes the address to be at 1FF0 to 1. Entry at 0320 assumes that the address is already in HL
0336	A programmable time delay. The computer loops through six E3s a long exchange instruction which, if used in pairs, does nothing but use up time. The number of loops is set by the byte immediately following the CALL in the original program. Each loop lasts 64 µs
0345	Clears the top line and sets DE at 8000
034E	Used to format results as in FIND and COR: this rounds DE up to the next multiple of 8
0355	The algorithm for encoding input commands. Returns with last letter of the command minus the first letter in register A
0372	The formatter used in LOAD and LIST in machine code language
0393	Clears the v d u leaving DE unaffected
039F	Displays HL and a space. Used in LIST, LOAD, FIND, COR and in BURP lists
03AA	Displays the contents of A as a two character hex byte
03C4	Calls a new line on the v d u and clears the remainder of the current one
03CE	Displays the string of characters following the call in the program block up to byte 1D
03DE	Loads HL from the keyboard
03E7	Loads A with a hex byte from the keyboard
03F6	Reads in a single keyboard character and, if a letter adds 6, then truncates to four bit binary (used as part of 03E7)

An exception to this is for the code FC, the routine for which 001E jumps immediately to 0042. This avoids one of the subroutines which have to be located at particular points in the memory map. Several subroutines can be called by single byte instructions which are known in mnemonic form as RSTs. These were originally intended for use with the 8080 and the Z80's "8080 mode" interrupt response which, after receiving the interrupt, calls for the interrupting device to place one or more instruction bytes onto the data bus for execution. Although this mode is not used, the single byte calls are a useful space-saver where a subroutine may be short and often needed. The subroutine which is avoided in this case at 0020 is called by byte E7 and produces a space on the v.d.u. At address 0028 is a jump to a subroutine which would require more than eight bytes. It is intended for use during the testing of machine code programs and when its RST byte EF is inserted into the program by using an ALT, it will suspend the execution of the program and display the contents of the HL, DE and AF registers, the point at which the EF was found, and the last entry onto the stack. Note that whilst there is not a specific subroutine at 0000 the one byte call CF for this address is used as an end command to a program. Although it does not do the same as C3 0000, because the stack is immediately re-defined at 0000, it has the same effect and saves two bytes.

Table 2. Machine code routine starting addresses

002F	FILL	0042	RUN	0040	MOV
0099	LOA0	00F7	PROM	0120	ALT
0113E	ALPH	014D	PRINT	016F	COR
01CD	FINO	0203	TAPE	0226	READ
01A6	LIST				

The two interrupts also use fixed service routines. At 0038 is the maskable interrupt routine which reads in and stores number cruncher data using HL as a pointer. At 0066 the non-maskable interrupt's routine services the keyboard, first checking if the computer is at a HALT byte (76) and reading in the keyboard if it is or reseting the computer if it is not (006B is an example of a long relative jump). This particular software does not make use of the control characters available in ASCII except for the RETURN byte 0D, which it translates to 1F. Instead, it blanks off the top three bits of any codes above 3F (mainly the letters) at 007C and moves lower and upper case codes into the area 00 to 1F. This compression of the ASCII code into six bits produces bytes which are compatible with the v.d.u. character generator and this makes writing to the v.d.u., which occurs at many places in the monitors, a simple operation.

Beyond the service routines, the routines for the various operations in table 2 fit end to end up to 0253, with the exception of some unprogrammed space at 0130-9. This space may be used by overprogramming the jump byte 011F-10 and the ten bytes as required. Note that the LIST (01A4) routine is just a call to a subroutine at 0317 because an identical block of instructions are required as part of the ALT routine. As this is the last command code to be checked, the call is conditional on a match so that if the code is undefined, execution passes to 01A9 and a software reset.

Table 3. BURP subroutines

0400	Used in graph plotting. Converts a number stored in 1E00-F to the nearest integral value. Negative values are put to zero.
042E	Executes MM57109 instruction present in register A. The sequence checks that the 57109 is ready, outputs the instruction with the hold off, waits for the ready to go off and then puts the hold on again.
0446	Repeats 042E for the string of 57109 instructions following the call in the main program. The list is terminated by FF.
0452	Jumps over the next word in the program line. Used in FOR statements to miss STEP and UNTIL.
045A	04E6 with aC protected.
0480	Outputs the contents of the 57109 X register to locations 1FF4 - 1FFF and then reformats it into the location specified by the contents of register A at the call i.e. 1E10 for 0F in A, 1E20 for 02 etc.
049A	Converts denary digits in the text to binary in register C. HL must be pointing to the first digit which must be in register A.
04D4	Graph plotting routine which scales the variablee to be plotted to the screen matrix of 83 × 126. It divides the variable specified by the contents of A by the declared maximum for that axis, and then multiplies by 126 before outputting to 1E00.
04E6	Jumps any spaces in the text and then analyses the following for (a) operators (04F8) which are converted by algorithm and 57109 code and executed, (b) numbers (050F) which are rearranged and then input to the 57109 (c) instructions (054E) which are encoded by algorithm and the result used as part of the address for the location in a look up table (positioned at the end of the rom) where the 57109 instruction code can be found, drawn and executed, (d) variables (0576) which are founded in 0460 and entered as a series of 57109 instructions. When encoding words the standard algorithm two times first plus last is used but to compress the range of codes produced those under 20 have 20 added and those above 50 are reduced by 10. This compressed byte is then added to 0784 which is used as a pointer to the instruction required. Some instructions need two bytes for their execution, the first being 20, e.g. 24 is SIN but 20 then 24 is SIN^{-1}. These are encoded in the table and detected at 0566 by bit 7 of the instruction being set.
058A	Handles the 57109 BR (branch) output which pulses low whenever one of the 57109 test instructions proves to be true. The subroutine starts the execution of the instruction in register A and then reads in the 6 bit data word from the 57109. The four digit out lines are blanked off so that only the READY and BR lines, both initially high, get through (0591). By continually reading and jumping back on even parity the Z80 is effectively waiting for one of these two lines to become active. If it is the BR line the Z80 outputs a NOP to the 57109 because, when tests prove true, the 57109 immediately looks for a new instruction and waits for its completion. If READY became active to indicate a failed test the last procedure is jumped. Finally, the read in and masked byte which caused the exit from the parity checking loop, stored in register B as part of that loop, is read into A and masked for bit 6 so that the state of the line and hence the zero flag in the F register is set on a successful test.
05A9	With the HL register pointing to a variable in the text, and with that variable in register A, this subroutine computes the variable's address, formats it into 6 bit ASCII, in the area 1E00-F and converts results in the range 0.000 to 99 999 999 to floating point. The byte in the text is checked and, if a digit, is used as the new mantissa digit count to be stored at 1FE0 (063A). Whether or not the contents of 1FE0 are then drawn out, the block from 064F to 068F rounds off the figures after the decimal point to the extent indicated by this digit. Blanked figures are replaced by ASCII spaces. The mantissa is then sent to the v.d.u. and the text interrogated again; this time for a comma which has the effect of suppressing the printing of any spaces and close packs the digits in the number (0693-7). Finally, at 06A3 an E for the exponent is looked for and if found the exponent is displayed. The alternative is three spaces or nothing, depending upon the comma for floating point numbers.
06BB	Prepares the store area specified by the contents of A using 0714 and then reads in a number from the keyboard, converting standard and non-standard scientific and floating point arrangements to the machines standard format.
07f4	Prepares a number store by dumping 9 00s, 60Fs and a 3F. This means that 06BB dump the input data into the store without having to worry about leading or trailing zeroes or the non existence of an exponent (0Fs being NOPs as well). The 3F terminates number entry to the 57109 as well as being a NOP therefore two consecutive variable inputs to the 57109 do not have to be separated by an ENTER as with reverse polish calculators.
0729	Algorithm for entering words from keyboard (two times first letter plus last).
0736	Inputs denary keyboard digits to binary in C.
074A	Converts A to three digit denary and displays on v.d.u.
0760	Converts A to three digit denary and displays on teleprinter.
07A2	Data for MOO command.
07AC	Forms the address for the start of a variable store area in HL from the variable code in A.
07B7	Displays a number formatted by 05A9 in 1E00-F displaying E for 0B, for 0A, a space for 0F, for 0C and ASCII digits for 00 to 09.
07D6	The look up table for the 57109 instructions.

Table 4. Format for storing and printing three variables

```
005 LET A=23 45
006 LET B=-0 00733
007 LET C=3 456E33
008 PRINT A
009 END
0C45

1E00  A0  32  33  2E  34  35  30  30  20  20  A0  A0  A0  A0  A0  6F
1E10  02  0A  03  04  05  00  00  00  00  0F  0B  0F  00  01  0F  3F
1E20  07  DA  03  03  00  00  00  00  00  0C  0B  0C  00  03  0F  3F
1E30  03  0A  04  05  06  00  00  00  00  0F  0B  0F  03  03  0F  3F
```

Table 5. BURP routine starting addresses

083F	DEL	08C7	MOD	0BF7	AOD
0929	LOAD	092F	LIST	0939	DUMP
0977	RUN				
099A	INPUT	09B2	PRINT	09D8	END
09DF	GOTO	09EB	LET	0A04	IF
0A43	WRITE	0A83	ERASE	0ABF	TOF
0A9E	GOSUB	0AAA	RETURN	0AB2	FOR
0AE2	NEXT	0B13	HALT		

From 0254 to 03FF are the subroutines listed in table 1. When necessary, the subroutines PUSH registers to be used solely within the subroutine and then POP them back before the return so that no interference is caused to data within the main program. Most subroutines are self-contained but some, e.g. 02EC, jump on to others for their completion. As subroutines are sometimes called within subroutines, within subroutines etc, the stack storage area, which extends into the r/w m from 1FDF, should be left free to at least 1FC0 for the computer's use. Like C7, other space savers will be found in the subroutines, e.g. AF to clear the register A instead of 3E 00, A7 to set the flags according to the contents of A. To save byte space some apparently unnecessary bytes appear, e.g. E3 at the start of the routine at 03C4 is included so that it and 03CE can share the same ending. Care is needed when writing subroutines because with a lot of PUSHing, POPping and EXchanging taking place it is important to ensure that the bytes called back off the stack by the return command at the end of a subroutine are definitely the correct ones. I have found this to be the Z80's most adept way of erasing painstakingly developed programs. This type of error is common when a conditional return is used as in 034E which prints spaces until the lower three bits of DE are zero. Ideally this should have pushed AF initially as it uses A and F, but to also arrange for them to be restored on return would extend the routine to at least nine bytes. The EF described earlier is a very powerful tool for sorting out these problems.

BURP monitor

For the BURP monitor the first p r o m is solely for subroutines whilst the second contains the operating system which makes use of them. Details of the subroutines are given in table 3. In BURP program material is loaded from 0C00 on, the area 1E00 to 1E0F is used for the formatting of results to be printed and 1E10-F stores variable A and so on up to 1FB0-F which holds the FOR loop step. Table 4 shows the formatting used for the storage and printing of three different variables. Note that all results are stored scientifically to maintain eight digit accuracy. Although the MM57109 can operate in either mode, it is quicker to stay in the scientific mode and let the Z80 convert the results within the range 0.0001-99999999 to floating point for display.

At 0800 the stack pointer is set and DE is assigned as the screen pointer again. BURP is then displayed and the rest of the top line cleared. The mantissa digit count is set at 04 (0817) and the screen position reset to 8008 ready for the input of a command. 081E to 0823 is harmless nonsense and 0824 to 0837 resets the number cruncher by sending the operation 3F (NO OP) with the hold to the 57109 off, pausing for 8ms and then reapplying the hold. During this sequence the interrupt mode is set but as it is the masked one that is driven by the number cruncher, the somewhat capricious behaviour of the i.c. before it has been reset has no effect upon the rest of the system. The i.c. is then given a master clear (2F) and switched to the scientific mode (22) by a multiple executive subroutine at 0446 (0832).

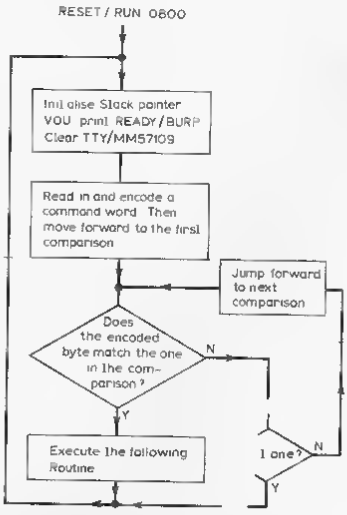


Fig. 2. Basic operating sequence for both monitors.

At 0838 another command encoder is called to read in a command from the keyboard. The algorithm used here is two times first plus last, so once again only two letters are required. However, this algorithm is capable of producing a far greater list of codes and therefore reduces the chance of two words deriving an identical one. As with the low level monitor routines entered by recognition of this code ensue, see table 5. The start of the last of these, for the RUN command, reads in and encodes the line number input in the command and stores it in register C. The v d u pointer is then set to 8040, the start of the second line, and C is decremented, pushed, popped, incremented and then pushed again. Four of these operations might seem to do nothing to C and on this occasion they do not. The total effect is to store the current line number on the stack. When the execution of a line is completed however, the next line number can be computed and saved by returning to 097F. After GOTOs, when A will hold the next line number, a pop to remove the old number followed by a jump to 0981 will load this as the next line to be executed. As all lines will terminate by jumping back to one of these locations (except for END which returns to 0800), to avoid absolute jumps (i.e. jumps to specific addresses), relative jumps to these two critical points are string out through the third p r o m.

A line of BURP is stored as the hex byte ED, the line number in hex, the actual data in modified ASCII and then the byte 1F to signify its end. The end of the memory block in use is signalled by the byte C0. With the commands ADD, DEL, DUMP, LIST and RUN involving line numbers the interpreter scans the program block up to C0 and looks for ED followed by the line number in question. During a RUN the next word in the line is encoded using the two times first plus last algorithm (0993) and again the routines for all of the commands are strung end to end and each is preceded by an immediate compare and a jump-on-not-zero (20 hex). The last command, HALT, compares at 0B0F and if a match is not made the computer jumps over the single byte 76 of the HALT routine and goes on to the next line by executing several relative jumps back to 097F. This explains why there is no routine for REM as it and any unrecognised first word on a line is just

Table 6. New features of the improved firmware

General
V d u cursor on all modes
DEL delete last character on all modes
RETURN available in graphics mode
Interface for ASR or KSR teleprinter (as printer and/or punch)

BURP
Extended IF statements Any statement may be conditioned by IF
Mathematical capability available in IF FOR PRINT, WRITE GRAPH and AXIS statements
Printed strings in INPUT as well as PRINT statements
Multiple statements — virtually unlimited numbers of statements may be written against a single line number
This speeds execution and expands the effective statement capacity well beyond the 254 lines

Extra maths functions

ABS	makes current result positive
INT	blanks digits following decimal point
FRAC	blanks digits preceding decimal point
RND	places pseudo random number into the MM57109

No need for LET at the start of LET type lines
In a line causes the computer to ignore the data following up to the end of that line (alternative to REM)

Hardware changes required
The wiring of several spare keys
The teleprinter interface shifted from O_7 to O_0 and 55V reduced to 5V

P r.o.m. required
Complete with the graph plotting firmware this will still fit into three 2708 e p r o m s

ignored (the very requirement for REM) Throughout the monitor the register pair HL address the program block contained from 0C00 onwards, whilst B and C are available for general use within the execution routines

Subroutine p r.o.m.

As far as possible subroutines have been written which can be called in many different places within the interpreter This particularly applies to 04E6 which can be thought of as a basic text handler which recognises and deals with words, variables, numbers and operators

In the next part a new monitor will be described, the features of which are given in table 6 I would like to thank all of the readers who have contacted me with suggestions for extra facilities and I hope that the new system will meet their requirements Lists of the new firmware will be available from *Wireless World* (editorial department) upon receipt of a large s a e. and these will be a useful accompaniment to the details in part two

The author is offering a set of three p.r o.ms programmed with the new monitor hardware for £30 Alternatively, existing p.r.o.ms can be reprogrammed for £6 50 (both plus 35p post and packing)

Micro show is bigger

Personal computers are prominent among the systems to be displayed and discussed at the Microsystems '80 conference and exhibition, January 30 to February 1. Sponsored by *Wireless World* and associated electronics and computer journals, this annual event has grown in size to such an extent that it has had to be moved from its hotel venue to the Wembley Conference Centre (opening hours, 0930 to 1800 hours each day)

The 1980 conference has a four-part programme ranging from an introduction to microprocessors to an overview of the latest developments in microelectronics Topics include technology update, micro processor software, controlling microprocessor projects, microprocessor applications, bridging the hardware/software gap, and microprocessors in process control The conference will concentrate on personal computers on its third day

There will be buyers' forum sessions to help people in selecting goods and services, and a one-day appreciation course to introduce managers to the use of microprocessors in business and industry Delegates' fee for the conference is £145.50, including v.a.t. and booking forms are obtainable from the organizers, Iliffe Promotions, Room 821, Dorset House, Stamford Street, London SE1 9LU (telephone 01-261 8113). The exhibition, with some 110 stands, is open to all at no charge, whether or not the visitor is a conference delegate

Literature Received

Reference sheets on the world's **electronics industry** produced by Mullard, showing exports, consumption, production of a variety of products. Sheets can be obtained from Mullard, Ltd, Mullard House, Torrington Place, London WC1E 7HD WW401

Leaflet on the ZIP KDP **computer terminal**, comprising 30ch/s dot-matrix printer, keyboard and v d u , can be had from Date Dynamics, Data House, Springfield Road, Heyes, Middx. WW402

Fourteenth edition of **Intel News** contains descriptions of an 8086 single-board computer, 1Mbit bubble memory and other items of interest in the computing field Intel Corp (UK) Ltd, 4 Between Towns Road, Cowley, Oxford OX4 3NB. WW403

Solid-state relay applications manual on specification, preotection circuits, loading and failure modes, with typical circuits, is available from Hamlin Electronics Europe Ltd, Diss, Norfolk IP22 3AY. WW404

Full ordering information on the component parts of the Elma **collet knob** range is available in broadsheet or wall-chart form from Radiatron Components Ltd, 76 Crown Road, Twickenham, Middx. WW405

Signal-conditioning amplifiers in the SE 990 series are described in a leaflet now available from Spur Road, Feltham, Middx, TW14 0TD WW406

Data for meteorologists, oceanographers, and ecologists can be collected by sensors on ships, without attention from the crew, collated by a **data collection platform** and transmitted to a satellite for retrieval The McMichael platform is briefly described in a new leaflet from McMichael Ltd, Wexham Road, Slough, Berks SL2 5EL. WW407

Leaflet from Astralux gives full details of 8000 series of opto-coupled, **solid-state relays** in 10, 20, 30 and 40A versions Sales department, Astralux Dynamics Ltd, Brightlingsea, Colchester, Essex CO7 0SW WW408

Selection of test equipment for **logic-testing** is presented by Electroplan in a four-page leaflet, obtainable from Electroplan Ltd, PO Box 19, Orchard Road, Royston, Herts SG8 5HH WW409

Various types of **panel mater**, counters, printers, etc., are described in a 48-page catalogue, produced by Techmation, Ltd, 58 Edgware Way, Edgware, Middx HA8 8JP WW410

Brochures on the Amercen Crown (Amcron) range of **audio equipment** can be had from the sole UK distributors, HHB PA Hire and Sales, Unit F, New Crescent Works, Nicoll Road, London NW10 9AX WW411

A collection of tools for **bending and cutting** component leads and for handling t.c. packages is detailed in a Wyber catalogue from Eraser International Ltd, Unit M, Portway Industrial Estate, Andover SP10 3LU WW412

More on the scientific computer — 2

An improved monitor

By J. H. Adams, M.Sc.

Since publication of the scientific computer, correspondents have suggested several features to improve the performance. This new monitor incorporates many of those features and includes a general expansion of the facilities available in BURP, including the routines for graph plotting. By restructuring the interpreter four extra functions, described in table 7, have been fitted into the three original e.p.r.o.ms. The demonstration programs have been removed, but these could be stored on tape, and the Creed 75 teleprinter interface has been replaced by a standard 110 baud ASR/KSR interface. The KSR machine is now cheaper and is fairly standard whereas the 75 may have different speeds and encoding as I suspect some readers have found to their cost.

Hardware modifications

Connections for the two extra keys are shown in Fig 3. The interface for the teleprinter is essentially a latch as in the original design, but this must be connected to D_6 instead of D_7. Most teleprinters contain an interface card for a 20mA loop or an RS-232 link. For a current loop, the second circuit drives the printer quite satisfactorily.

Firmware modifications

Changes to the firmware are detailed in tables 8 and 9. Primarily, space has been made in the first e.p.r.o.m. for three of the subroutines originally in the second which deal with instruction entry and condition testing of the MM57109. This has been achieved by using a simpler and shorter teleprinter interface, eliminating the subroutine at 034E, and trimming the low level monitor so that it ends at 024E. This has left space in the second e.p.r.o.m. for a new subroutine 051D which extends the old 04E6, now 047C, and together they can recognise and deal with the new facilities. Because these routines are quite complex, a disassembled listing of each is given in table 10.

The third r.o.m. is slightly briefer because checks for ends of lines, present in virtually all of the statement handling routines, are replaced by 051D. The command MOD (08BE) has been changed so that PRINTs buried in multi-statement lines are also changed to WRITEs. CALLs have been readdressed to suit the first two r.o.ms and CALL 042E has been replaced by the single RST byte CF (see 0008). In the original r.o.m., after going through the sequence of recognition checks for encoded commands or, later, first words of statements, the interpreter returns to the command state or ignores the rest of the line respectively, if it cannot find a match or the generated code within the firmware.

This is particularly useful for dealing with REM because, being unrecognised, such lines are ignored as explained last month. A major change in the modified r.o.m. provides jumps to 1C00 (at 0975) for commands, to 1C60 (at 0AD7) for new statements and to 1D00 (at 0BDE) for new functions. As a result REM has disappeared but the apostrophe has the same effect and retains the facility for remarks.

0993 is an example of where 051D is used solely to jump spaces between the line number and the first word of the statement. Therefore, it is the point to which 051D transfers execution after coming across an 1 in the text being interpreted. 097F pops off the stack, increments and pushes back the C register which is used as the line register store and then looks for and executes that new line. Thus, it is the point to which 051D transfers control after finding a ' or 8DH number in the text. Because the computer scans the text for line numbers whether they exist or not, the lines in a program should be as close together as possible (say every other line) for the fastest program execution. Using multiple statements avoids this problem to some extent and can therefore reduce the execution time of some programs, particularly simple ones, by up to 20%.

Table 7. Additional facilities for the new monitor

INT (0B64)	Outputs the number in the 57109 to 1E00 — F and tests the exponent sign. If negative, the whole number is written to zero; if positive, the lower mantissa exponent is drawn and used to calculate (0B72-B) where blanking should start. If the exponent is not less the 09 (0B80 B) blanking is carried out. The number stack in the 57109 is then collapsed by one to remove the old value (0B97) and the new value is entered into the 57109 by a jump to 050F at 0B9A.
FRAC (0BA1)	Outputs the number and tests as in INT. If the exponent sign is negative execution jumps to 0B96 (0BA5) and effectively does nothing. For positive exponents a similar sum involving the lower mantissa exponent digit is performed and a jump is made back to 0B79 in the INT routine (0BAE).
RND (0BB4)	029F is called which loads the refresh register into A, converts it to a three digit decimal integer and enters it into the 57109 (this subroutine runs straight into 02AD). A pseudo-random delay (0BBB-A) based on the current v.d.u. printing position is then called so that a second call of 029F will generate a second number from the ZB0 refresh register which is only tenuously linked to the first. These numbers, now in the Y and X registers of the 57109, are combined through the sequence of instructions at 0BBE to give X = 12BX + Y/163B3, i.e. a reasonably random number between 0 and 1. Note that as this uses two of the 57109 stack registers, no more than two other variables must be present in the 57109 when RND is used.
ABS (0BD3)	This simply uses the number cruncher test instruction 12 to test for a negative number in the X register. The result of this test governs whether the instruction to change sign, OC, is executed.

Table 8. Alterations to the first r.o.m.

024F was 03CE	0263 was 0260	02B2 was 058A
02A0 was 024E	02C7 was 0446	0326 was 0317
0345 was 0336	0367 was 0729	0374 was 0372
0395 was 0393	03A1 was 039F	03AB was 03A9
03C6 was 03C4	0301 was 0260	

029F Generates a 7-bit pseudo random number and inputs it to the 57109
02D1 Converts the computer 6-bit ASCII to true ASCII and prints it
0209 Prints a space
02DE Prints carriage return and line feed
02EB Prints the contents of register A
02F0 Prints (A) as a two character hexadecimal byte
0317 Prints CR, LF, the contents of HL in hexadecimal and a space

Using the new facilities

In low level the first feature to be noted is that READY does not disappear when a command is typed in nor does the first letter appear at the beginning of the second v d u line This is because the same algorithm is now used for both high and low level word recognition Clashes produced in the changeover explain the changes of COR to MOD and PROM to PROG To leave LOAD, the space key is now used instead of @ The main change which affects both levels is that the interrupt-and-reset, which occurred whenever any key was depressed, has been omitted because control can be regained by using RESET The "arrow" keys now revert to standard keys, RESET enters the low level and Control A (depressing A and the control key simultaneously) enters the high level The delete key to the right of] can be used to delete complete bytes by one depression per byte Although this will cause the formatting to go out of true during the LOAD, the grouping by four is maintained and on pressing the space bar at the end of the load the format will be restored

When loading programs in high level language, another character Control E is used to signify the end of LOADing or ADDing This allows the colon, which was previously used for this purpose, to be included in printed messages etc without terminating the current operation Ensuring correct format of the input has been eased by a cursor, although with the original monitors few problems will be encountered if a space is typed when in doubt The DEL key backsteps and clears the last v d u character and also backsteps HL Corrections are, therefore, easily typed in, but mistaken returns and line numbers cannot be corrected in this way because

Fig. 3 Modifications to the keyboard and teleprinter interface

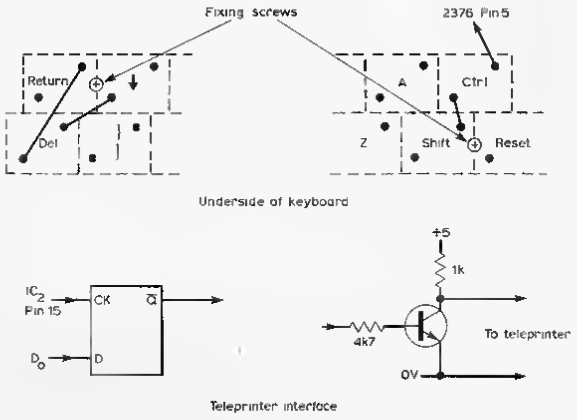


Table 9 Firmware changes

0400	Old 04D4 running straight into 040D
040D	Old 0460
0467	Old 048A
047C	Old 04E6. 04FA-E is added to this so that when a code of less then 08 is drawn from the look up table at the end of the r o m execution jumps to 0860 These new codes are for ABS, FRAC, INT RND and any others which are not simple MM57109 operations and will thus require some Z80 software
051D	Jumps spaces and then returns on bytes less than 18 and greater or equal to 2A (except for 8D) Thus for letters operators and spaces this routine will just jump spaces and return with HL pointing to the first non space. i e 051D is a supplement to 047C If the byte found lies between 1A and 2A it will after (a) (052D) transfer text up to the next onto the v d u and then jump back to the start of the subroutine to deal with whatever follows (b)) (053B) collapse the stack and return (c) ((0542) call 051D to jump spaces and then 047C to execute the text within the parentheses until the call of 051D finds a) As this) will have been found during the calling of 051D at 0546 and as) indicates that the original call of 051D is no longer required. i.e the bracketed term has been computed detection of) drops the stack pointer past the return address the call at 0546 so that a return is made to the original point in the interpreter from where 051D was called After dealing with an expression in perentheses the computed result is left in the X register of the 57109 and the SCII for) 29 is left in register A If the interpreter has not yet recognised the byte it must now be at the end of the statement Before looking for a ' or 8DH two types of statement need special attention 1FE1 is used in the third r o m (0999) to store the code generated from the first word of the line If it is 33 (i e a WRITE statement) execution shifts from 0554 to 056B WRITE lines are similar to print types except that the material to be displayed is fed to locations from 1D80 rather than to the v d u 0568 sets an FF at the end of the block used and then resets DE to 1D80 and outputs the characters up to FF on the teleprinter After restoring AF and DE it returns to 0563 If the line is a LET (code 2C) the variable to which the computed value is to be assigned is drawn from its store (1FE2) and the contents of the 57109 X register are fed to it After dealing with these two special cases checking of the original byte continues (0560) The remaining possibilities will transfer control rather then return from the subroutine and so the pointer is moved down the stack losing the previously stored return address and then after (d) ' (0563) execution passes to 0993 (e) 8DH ' or anything else passes execution to 097F 8D is the code for return and indicates the end of a line signifies that the rest of the line is a remark which the interpreter will also want to treat as the end of a line
0582 4	Jumps text and then calls 051D and when required (i e letters operators or digits) 047C as well
0589	Calls 051D as above
0594	Old 0714
05A9	Unchanged
0736	Unchanged
074A	Modified 074A
075A	Old 076D
0773	Used in the above two to cover common parts and thus save space
0789	Used in INT and FRAC
07A2	Unchanged
07AC	Unchanged
07B7	Unchanged
07D6	Look up table which now includes codes for new functions (07DA/DC/E3/E9)

they involve internal operations by the interpreter rather than the byte by byte storage which takes place during lines. The critical formatting points are LET lines where the variable following let must be followed immediately by the equal sign, and IF lines where, when a variable precedes the comparison sign, there must be a space in between.

A program in table II demonstrates the uses of the new facilities. Lines 3 and 4 show the new REM and in this case they are complete lines on their own. Remarks may be appended to any "active" line just preceded by an apostrophe. Line 5 shows printed text in an INPUT line. The input variable X is against the " to save r/w m. space but again, spacing is not critical. In line 7, two spaces are left between step and I without any effect on the interpreting of the line. Note that the expression in parenthesis is spaced exactly as in a LET statement. Line 9 demonstrates the compounding of two LET type statements (with the LET omitted) by the use of an exclamation mark. The statement following ! is typed immediately after the !, again to conserve r/w m. space. Line 11 is "If K is a whole number and if Z is also a whole number, then print half of K plus A to two decimal figures and then half of the positive difference between K and A". This line illustrates the need for a space between the variable and the greater than, equals or less than sign. A space is required because, under the original interpreter, this had to be a variable but it can now be a variable, number or function in parenthesis and therefore has to be distinguishable. A closing parenthesis has no other meaning and does not need the space, i.e. IF (X SIN I —)=Q print . .

The text following an IF comparison can be any other permitted statement including another IF as shown in the example program. Therefore, the old form IF X=0 THEN 125 will be IF X =0 GO 125. It might seem that the freedom to place statements end to end on the same line will reduce all programs to one line in length (note that a line is not determined by the length of a v.d.u. line and may consist of any number of characters). However, this is not so because whenever a statement has to be entered as the result of a jump, or it initiates a specific jump, the statement must either start or end a program line respectively. This means that the first instruction in a FOR loop must be at the beginning of a line because further through the execution a NEXT will try to jump back to it. Similarly, the statement after the complete IF term must be on a new line because IF is basically "perform the operation specified after the conditional test if the latter is true or jump to the next line"

By similar reasoning, GOSUB and GO should be at the end of lines, as should RETURN and END. The lines to which GOSUB and GO refer should start with the statement to which the jump was directed.

While encoding the new functions by algorithm, several clashes occurred with already assigned codes and this provided an opportunity to re-encode the two log functions into a more standard format, i.e. CLG for a common log and LOG for log to the base e. The radian to degree conversions have also been changed by dropping the first letter, i.e. TD for a conversion to degrees and TR for one to radians.

The author is offering a set of three p.r.o.ms programmed with the new monitor firmware for £30. Alternatively, existing p.r.o.ms can be reprogrammed for £6.50 (both plus 35p post and packing). 5 The Close, Radlett, Hertfordshire.

Table 10 Disassemblad subroutinas

```
047C  LD A,(HL)
047D  INC HL
047E  LD C,0F
0480  CP  20
0482  JPZ  F5 047C
0484  CP  1B
0486  JPC  51 04D9
0455  CP  30
045A  JPNC 17 04A3
045C  CP  2D
045E  JRNZ 03 0493
0490  CP  (HL)
0491  JPC  0C 049F
0493  ADD 0D
0495  AND A
0496  JP PE049D
0499  DB  09
049B  AND F8
049D  RST 1
049E  RET
049F  LD C,0C
04A1  LD A,(HL)
04A2  INC HL
04A3  EX AF AF'
04A4  PUSH DE
04A5  EX DE,HL
04A6  LD HL,1E00
04A9  LD B,0F
04AB  CALL 05AC
04AE  LD L,09
04B0  LD (HL),C
04B1  LD L,0A
04B3  EX AF AF'
04B4  AND 0F
04B6  CP  0E
04B5  JRNZ 02 04BC
04BA  LD A,0A
04BC  LD (HL),A
04BD  LD A,(DE)
04BE  INC HL
04BF  INC DE
04C0  CP  28
04C2  JRNC F0 04B4
04C4  CP  20
04C6  JRZ  44 050C
04C8  LD L,0A
04CA  LD (HL),2E
04CC  INC HL
04CD  LD A,(DE)
04CE  INC DE
04CF  CP  2D
04D1  JRNZ E1 04B4
04D3  LD (HL),0C
04D5  INC HL
04D6  LD A,(HL)
04D7  JR  DB 04B4
04D9  EX AF AF'
04DA  LD A,20
04DC  CP  (HL)
04DD  JPNZ 03 04E2
04DF  EX AF AF'
04E0  JR  2E 0510
04E2  DEC HL
04E3  CALL 0715
04E6  CP  2C
04E8  JPNC 22 04EC
04EA  ADD 28
04EC  CP  50
04EE  JRC  C2 04F2
04F0  SUB 10
04F2  ADD B4
04F4  PUSH BC
04F5  LD C,A
04F6  LD B,07
04F8  LD A,(BC)
04F9  POP BC
04FA  CP  0B
04FC  JP C 0360
04FF  CP  80
0501  JRC  05 0503
0503  EX AF AF'
0504  LD A,2A
0506  RST 1
0507  EX AF AF'
0508  AND 3F
050A  RST 1
0508  RET
050C  DEC DE
050D  EX DE,HL
050E  POP DE
050F  XOR A
0510  PUSH HL
0511  CALL 07AC
0514  LD B,10
0516  LD A,(HL)
0517  INC HL
0518  RST 1
0519  DJNZ  FB 0516
051B  POP HL
051C  RET
051D  INC HL
051E  LD A,(HL)
051F  CP  28
0521  JRZ  FA 051D
0523  CP  1B
0525  RET C
0526  CP  3D
0528  JRZ  03 052D
052A  CP  2A
052C  RET NC
052D  CP  22
052F  JRNZ 0A 053B
0531  INC HL
0532  LD A,(HL)
0533  CP  22
0535  JRZ  E6 051D
0537  LD (DE),A
0538  INC DE
0539  JR  F4 0531
053B  CP  29
053D  JRNZ 03 0542
053F  INC SP
0540  INC SP
0541  RET
0542  CP  28
0544  JRNZ A8 04EE
0546  CALL 051D
0549  CALL 047C
054C  JR  F8 0546
054E  PUSH AF
054F  LD A,(1FE1)
0552  CP  33
0554  JRZ  15 056B
0556  CP  2C
0558  JRNZ 06 0560
055A  LD A,(1FE2)
055D  CALL 04CD
0560  POP AF
0561  INC SP
0562  INC SP
0563  CP  21
0565  JP Z 0993
0568  JP 097F
056B  LD A,FF
056D  LD (DE),A
056E  LD E,80
0570  LD A,(DE)
0571  CP  FF
0573  JRZ  0B 057D
0575  AND 3F
0577  CALL 02D1
057A  INC DE
057B  JR  F3 0576
057D  POP AF
057E  POP DE
057F  POP DE
0580  JR  E1 0563
```

Table 11 Demonstration programs

```
003 'THIS PROGRAM, PUBLISHED IN PART 4, TOOK 19 LINES BEFORE. NOW...
005 PRINT "THIS PROGRAM USES NEWTON'S METHOD FOR SOLVING"
007 INPUT "F = F(X). ENTER AN INITIAL VALUE NOW "Q :ERASE
009 X=0 :GOSUB 25
011 G=F :IX=X :X=X-.00001 * :GOSUB 25
013 TOP !IF (G ABS )<0.00000! PRINT "SOLUTION ="G4 :END
015 Q=( F G / ( - REC 0.00001 * - Q * :PRINT @3 :GO 9
025 F=X LOG X 3 * + 10.3074 -
027 RETURN

0D78


003 'THIS PROGRAM COMPUTES PAIRS OF NUMBERS WHICH, WHEN
004 'SQUARED AND SUBTRACTED, GIVE THE INPUT NUMBER
005 INPUT "INPUT NUMBER IN QUESTION "X
007 FOR A=1 STEP  1 UNTIL (X ROOT 1 + )
009 (=X A / I2=X A - 2 / ABS
011 IF ( =(( INT ) IF I =(I INT ) PRINT (X A + 2 / )2 (( A - 2 / ABS )
013 NEXT A :GO 5

0D26
```

Floppy disc system for the scientific computer — 1

8in disc stores 400K bytes

by J. H. Adams, B.Sc. M.Sc.

Storage of data in small computer systems is often accomplished by a 300-baud cassette tape recording. With a transfer rate of only 2K-bytes per minute, this method makes locating and transferring long strings of data a rather slow process. The introduction, through the users' club[1], of a more advanced operating system for the computer[2], and the availability of memory expansion kits, has made a faster store very desirable. To solve this problem the author has developed a store based on an 8in flexible (floppy) disc, which can accommodate 400K-bytes of data and transfer 0.5K-bytes per second.

Recordings on disc are made by converting the data bytes into a serial stream of 1s and 0s at a rate of 250,000 bits per second, i.e. one bit every 4μs, truncating the 1s down to about 0.5μs pulses and then interleaving a regular stream of 0.5μs pulses from the system clock as shown in Fig. 1. These pulses are used to reverse the current in the recording head and, hence, the sign of the flux recorded onto the disc. Converting the parallel input data to a stream of pulses is most easily achieved by one of the controller i.cs which are, in essence, dedicated microprocessors combined with programmed logic arrays to feed control information between the controller, the disc-drive electronics and the computer. Recordings are made on concentric rings, or tracks, 77 on an 8in disc, and a drive unit with two motors rotates the disc and steps the combined record/read/erase head from track to track. Optical devices provide signals which indicate when the head is over the outermost track 0 and, using a small hole punched in the disc, an index pulse to indicate when the disc starts each revolution.

The electronics in the drive unit convert t.t.l. levels to switching currents in the head and vice versa, operate the stepping motor and provide erasing signals. Other functions may include door locking, motor-on indication, adjustment of recording current on inner tracks, separation of data, disabling the write operation on write-protected discs, and loading the head against the disc on read operations. This drive unit contains most of these features, although separation of data is achieved in the controller i.c.

Recording format

At 360 r.p.m. it is possible to record over 5000 bytes of data on each track of the disk. To allow the controller to identify recorded clock pulses from serial data pulses from the disc, the start of the decoding process is triggered by the index pulse, and the recording begins with a standard code which the controller can recognise and synchronize with. This code is often produced by repetitive recording of the byte 00, i.e. the clock pulses are recorded with no interleaved data pulses. The next task for the controller is recognition of the start of the first byte in the data stream. As all possible data bytes may start the stream, no single byte can be reserved for this purpose. Instead, a data byte with a few of the clock pulses missing is used and is known as a mark byte. Normal bytes can be thought of as data bytes interleaved with the clock byte FF, i.e. all eight pulses. A typical mark byte is data byte FC, interleaved by the clock byte C7. After this index mark, about 5000 bytes of data follow and the recording runs to the start of the next index pulse with a final code of bytes, usually 00s or FFs. The total number of code bytes is determined by the accuracy of the clock and drive motor.

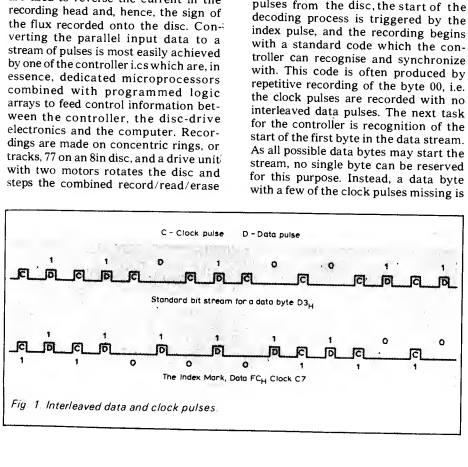


Fig. 1. Interleaved data and clock pulses.

Sectored tracks

If data transfers, which match the above, are all that is required of the disc, it is an efficient way of using the system in terms of bytes stored per disc. Usually, however, transfers are of variable length and, as it is not directly possible to access part of the way through a track, there is a limit of one recording per track, no matter how short the data block. To improve the potential disc capacity, each track is split into sectors which each require start and stop codes and identification marks. This leaves less space for data, but normally provides the most efficient mode. Such a format, now widely in use, is the IBM 3740 which fits 26 data sectors into each track, with 128 data bytes in each sector as illustrated in table 1.

In the present format, sectors consist of an identifying block followed by the data. Six 00s synchronize the clock/data separator, an address mark (data FE, clock C7) indicates the boundaries of the bytes, and, as previously explained, track and sector numbers are given. This is followed by a CRC, which is a two-byte cyclic recognition code used by the controller to check for errors when reading information. The sector then has a short code, immediately followed by six more 00s, a data mark (data FB, clock C7), the 128 bytes of data, a two byte CRC for the data, and a final code. Each track has 26 of these sectors end to end, prefixed by a large block of 00s, an index mark (data FC, clock C7), and trailed to the end of the track by a code. The copious supply of synchronizing bytes and CRC codes can, with suitable software in the computer, produce a very reliable system.

Formatting all of this information onto the disc is a complicated operation,

IBM 3740 sector

6 bytes 00	Ident mark	Track no.	00	Sector no.	00	CRC 2 bytes	17 bytes 00	Data mark	Data 128 bytes	CRC 2 bytes	27 bytes FF
Ident field							Data field				

IBM 3740 track

40 bytes FF	6 bytes 00	Index mark	26 bytes FF	26 sectors	approx. 240 bytes FF

Table 1. Formatted disc arrangement.

and most discs are supplied formatted with dummy data (usually byte E5). Such discs are marked 128 bytes per sector or per record, soft sectored. The last mentioned term means that the start of the sectors is indicated by software recorded onto the disc, as opposed to permanent hard sectoring, achieved by punching index holes in the disc for each sector (sometimes called 33 hole media for this reason). One disadvantage of soft sectoring is that, if formatting information gets magnetically corrupted, the sectors affected become useless. For this reason, even unused discs should be treated with care. Fortunately, if this should happen, the controller can re-format tracks to this and a number of other formats.

Computer-controller interface

To the computer, the controller looks like four input and four output ports. However, to address the controller, the computer only needs to supply one line each from IC_2 and IC_3, (the computer's input and output-port decoders) along with address lines A_3 and A_4 as shown in Fig. 2. Because neither of the address lines go to the decoding i.cs, they take no part in decoding the 8-bit port addresses which the Z80 sends along the bottom eight address lines during I/O instructions. Therefore, I/O commands such as IN(05), IN(0D), IN(15) and IN(1D) will activate the same line from IC_2, the bottom three bits of each number being the same, 101, but each provides a different combination on A_3 and A_4. By connecting the IN line to the controller's $\overline{RE}$ (read enable) input, and the two address lines to the A_0 and A_1 inputs, all four controller registers may be read by the computer. In a similar fashion, one line from IC_3 drives the $\overline{WE}$ (write enable) line of the controller, which allows the computer to write information into any of the registers. For details of these see Table 2.

As well as the data bus into and out of the device, and the four control lines described above, there are two lines from the controller to the computer. One indicates that, either through natural completion or through a failure, the controller has finished an operation and wants servicing, $\overline{INTRQ}$, the other, $\overline{DRQ}$, indicates that the controller desires a data transfer either to or from it. This information is present in the status register but, because of the high rates of data transfer taking place, these lines must be used to enter the Z80, through the interrupt line, in preference to the much slower polling of the status register. The Z80 can therefore keep up with the steady demand for, or supply of data between it and the controller. For this reason, part of the interface consists of a simple but effective interrupt controller.

Controller disc-drive interface

Lines from the controller to the drive comprise step and direction signals for the head-position motor, data and gating signals for the write operation,

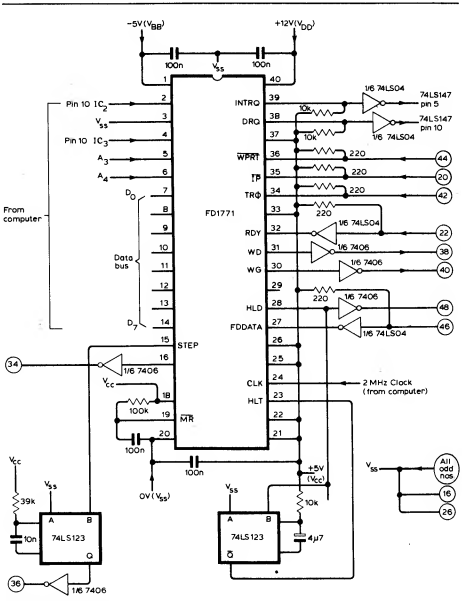


Fig. 2. Floppy-disc controller / formatter.

A3	A4	Register	Addressed as	Remarks
0	0	Status	IN A,(05)	Read on $\overline{\text{INTRQ}}$, checked for CRC error and record not found bits during Read, Write and Seek.
		Command	OUT (A0),A	Receives commands from the computer.
0	1	Track	IN A,(0D) OUT (A8),A	This register normally contains the current head position, 00 to 4C. It is reset to 00 on completion of the Restore command.
1	0	Sector	IN A,(15) OUT (B0),A	This register holds the desired sector number for use during Read and Write operations
1	1	Data	IN A,(1D) OUT (B8),A	Used during Read and Write operations as the source and destination of data bytes. During Seeks, it holds the desired track number, towards which the head and the track-register step.

Table 2. Register structure of the floppy-disc controller. Note that these details refer to the controller in this interface. More details are given in the data sheet[3].

and a head loading signal. In the opposite direction, the drive provides data in the form of interleaved clock and data pulses, the index pulse and signals to indicate that the head is over track zero, the disc drive is ready for use, and whether the disc is write-protected. Some drives provide a head-loaded signal, or can load the head onto the disc in the 10ms delay provided internally by the controller. As this drive does not, a monostable is used to provide a delay signal which is triggered by the outgoing headload line. To improve noise immunity when transferring signals to and from the electromechanical drive, the lines at both ends are pulled high via low value resistors, and high-current sinking buffers are used to drive signals to and fro.

Interrupt controller

Interrupts allow external hardware to divert the microprocessor temporarily from its stream of instructions, and accept instructions from, or more usually, to call a block of instructions which deal with the hardware's needs. In the computer, the $\overline{\text{INT}}$ line to the Z80 is driven by the MM57109 which only requires the Z80 to read (for Mk I and II systems) or transfer (for Mk III), data between the two. This is a fairly simple procedure and the interrupt mode is chosen, which causes a call to the address 0038 where, in the earlier two systems, there is a short routine to read the data. In the Mk III, there is a re-enable of the interrupt and return so that the interrupt line makes the Z80 pause until the MM57109 is ready for a data transfer. With the disc, faster and more complex responses are required because either of the two interrupting lines can become active separately or simultaneously and, depending upon the operation in hand, different responses may be required. To provide the extra flexibility without sacrificing speed, Mode 2 interrupts must be possible, and this is carried out by the interrupt controller in Fig. 3.

In Mode 2, after the Z80 has completed its existing operation, it responds to the interrupt by asking the interrupting device to supply an 8-bit byte onto the data lines which it uses, in conjunction with the previously loaded I register, to form the address of the first of two consecutive memory locations where it will find the starting address of a subroutine to be executed. With these two locations in the r/w.m., the disc operating system can make alterations to their contents and so alter the Z80's response to, in particular, a DRQ interrupt to cope with read, write and verify operations which are at the heart of a disc system.

The circuit uses a 74LS147 priority encoder to generate a 4-bit code derived from the highest priority active input line. This code forms bits 1 to 4 of the byte which, when both $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are active low, (a combination which only occurs during the Z80 interrupt acknowledge sequence) is gated onto the data lines. Direct connection to the Z80 is necessary because, during the interrupt response, both $\overline{\text{RD}}$ and $\overline{\text{WR}}$ are inactive and the bus transceiver isolates the Z80 from the main data bus.

The connection from the MM57109 to the $\overline{\text{INT}}$ pin of the Z80 must be broken and re-routed through one of the unused inputs to the interrupt controller. As the 57109 is now driving a low-power Schottky device, the pull-down resistor on this line must be changed to 10kΩ connected to the −5V supply.

Part 2 will describe the controller circuit and software.

References

1. Users' club, contact Mr P. L. Probetts, 50 Cromwell Road, Wimbledon, London SW19 8LZ.

2. Mk III monitor, contact the author at 5 The Close, Radlett, Hertfordshire WD7 8HA (Radlett 5723).

3. Data sheet, Western Digital FD 1771, Mar 77. □

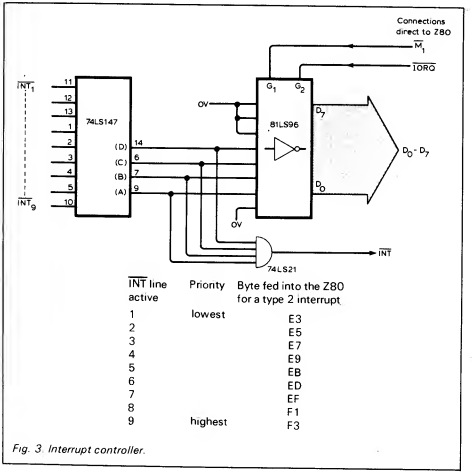


Fig. 3. Interrupt controller.

Floppy disc system for the scientific computer — 2

Interfacing a disc drive to the controller

by J. H. Adams, B.Sc., M.Sc.

This interface has been designed to operate with the Data Recording Equipment model 7100 8in disc drive, but should be easily adapted to suit others. The main advantage of an 8in drive over a 5¼in system is its greater storage capacity, 77 tracks of 3¼Kbytes each using the IBM format described in part one, compared with 35 tracks of 2¼Kbytes each. The disadvantage is greater cost. This concluding article describes how the drive is matched to the floppy-disc controller, and illustrates the salient points to check when considering other drives.

Whichever drive is used, the length of the cable, flat or twisted pairs, between the drive and the interface must be kept as short as is reasonably possible, and separate from the power cables. Each power cable should have its own return and there must be a good connection between the frame of the drive and the case of the computer.

When considering the signals to and from the drive, their polarity and timing must be examined. Most drives use the active-low principle for their inputs and outputs, i.e. a true state is logical zero (represented by $0 \leq V \leq 0.4V$) and a false input is logical 1 (represented by $2.5V < V < 5.25V$). Open collector drivers are generally used for outputs, and low value pull-up resistors on the inputs provide a full 5V swing and keep the line impedance down, both of which improve noise immunity. One implication of this arrangement is that, to pull a line to zero, the driving device will need to sink the current supplied by the receiving gate and by the pull-up resistor, typically 40mA. For a logical 1, no current is required from the driving device. The controller i.c. signals are mostly active-high, so inverters are used as receivers on all inputs except $\overline{IP}$ (index pulse) and $\overline{WPRT}$ (write protect), and 220Ω pull-up resistors are used with high current-sinking, inverting, open-collector drivers on the five active outputs. If a drive with some active-high inputs is used, the equivalent non-inverting buffers, 7407, or pairs of 7406 in series must be used. Note that ordinary t.t.l. is used for driving the interface cable because the L and LS series do not have the required current-sinking capacity. Table 3 gives some timing information for the 5¼in drive, the 7100 and the WD1771 controller.

When used with an 8in unit, the WD1771 must be clocked at 2MHz, whilst with a 5¼in disc 1MHz is used. This is necessary to meet the standard data rates used for the two sizes, and results in the doubling of all pulse timings for the i.c. when used with the smaller disc. There are other timing requirements connected with the application of power and selection of the drive, but these can be allowed for in the programming of the computer.

Stepping time

Most drives offer the option of keeping the head permanently loaded against the disc. This speeds operations by eliminating head-loading delay, but does increase wear on the head and disc. In this operating system, the head is only loaded when necessary which is usually after it has stepped to the track required. Settling time is irrelevant if it is less than the head loading time. The interval between stepping pulses is programmable by the bottom two bits of the Step instruction byte as described on page four of the data sheet. With the 7100 drive the fastest (6ms) rate can be used, whereas with the 5¼in unit, the drive can only just keep up with the slowest stepping rate. Stepping rate is probably the most critical timing factor because if a drive cannot step as fast as the controller's slowest rate, the two are virtually incompatible.

Stepping pulse

Virtually all drives can step on the pulse provided by the 1771. However, one exception found by the author is an obsolete version of the 7100. As this unit may still be available, the interface has been designed to operate with it and the current version.

The obsolete 7100 is most easily recognised by the absence of three d.i.l. sockets and header plugs next to the edge connector on the p.c.b., and the presence of two power resistors and three power transistors instead of one resistor and four transistors near the opposite corner to the edge connector. To allow compatibility, a monostable stretches the stepping pulse to 10μs.

Head-loading time

Ten milliseconds after the HLD (Head Load) output of the controller becomes active (20ms for the 5¼in disc), the HLT (Head loaded test) input is sampled and when it is low the controller proceeds. If the combined loading and settling time for the head is less than 10ms, this input can be wired low. If the disc-drive electronics provide a head-loaded and ready signal, this can be connected to HLT. If neither is true, as it is for both of these drives, HLD should trigger a monostable to produce the necessary delay before HLT becomes low. Because most drives will need this monostable if they are to be used with the head normally un-loaded, it is important to establish the total delay before the head is ready for use, i.e. the loading and settling time. Note that one value cannot be inferred from the other by comparing the stepping rate figures for the two drives.

Drive options

Most drives offer wiring options, and in this system one for direct control of head loading by the controller is used. To select this option on the current model, remove the link joining pins 13 and 14 on plug PP2 (the middle one of three referred to earlier).

Table 3. Timing information for disc drives and the controller

	5¼in drive	WD1771 @ 1MHz	DRE 7100 8in drive	WD1771 @ 2MHz
Track to track stepping + settling times	40 + 10ms	programmable to 12, 20 or 40ms	4 + 14ms	programmable to 6, 10 or 20ms
Stepping pulse width	1μs min.	8μs	600ns (10μs on older units min.)	4μs
Head load and settling time	75ms	HLT sampled after 20ms, therefore monostable is required	30ms	HLT sampled after 10ms, therefore monostable is required

and join pins 3 and 14 together. On the obsolete version, remove the short wire link joining the points marked HL and SI and connect a wire from HL to the pad at the end of the p.c.b. track coming from the edge connector tab numbered 18.

This change allows the controller to drive the head-load circuit through the previously unused pin 18 on the edge connector. The current and obsolete units should now be interchangeable.

System software

The software in table 4 is not a full disc operating system, but it illustrates the basic functions required to position the head, read and write records of any length from 128bytes to 256Kbytes with error checking, and to re-format corrupted tracks. With the drive and interface connected to the computer, move the head towards the centre of the drive by turning the stepping motor by hand. Apply mains to the motor and then switch on the computer. Put a disc into the unit and close the door. If the system is working, the head should quickly step to the outermost track 00 because the charging delay, caused by the RC network on pin 19 of the controller i.c., holds that pin (master reset) momentarily low in a similar way to the circuit used on the Z80. One of the actions which takes place during the resetting sequence is a Restore command, which moves the head out in this way. If it doesn't, check that the wiring is correct. If it steps out but does not stop, check that the track 00 line from the drive to the controller functions. With the software loaded, RUN 1D00 and then READ space. In response to the prompts DESTINATION, TRACK, SECTOR, NUMBER OF SECTORS, type 8200 40 space 0 space 8 space respectively. The head should move in to track 40 and load 8 sectors (1Kbyte) of data from the disc, starting with sector 0 to computer locations 8200 to 85FF, i.e. onto the v.d.u. With the IBM formatted disc, these should appear as percentage or proportional symbols.

At the end of the read, which should take less than one second, the head should release from the disc and READY occur. If a reading error occurs, the computer will attempt to re-read the particular sector up to twenty times. A corruption should be evident by rubbish appearing on the v.d.u., the controller recognises it by computing the CRC from a permutation of the data from the sector and comparing it with the pre-recorded CRC. Each sector takes up two lines on the v.d.u. If the corruption begins in a line and keeps changing, the data is corrupted. If the reading process seems to stop at the end of a line, the controller is having trouble recognising the Ident Field for the track or sector and, therefore, the track needs to be re-formatted (described later). With an undamaged disc most reads are successful first time, but if the operation fails for the 20 times that it is attempted, the message ERROR AT TRACK XXX SECTOR XXX appears. To force an error into the system and observe this feature, try

```
1D00 ED 5E 3E 1D   ED 47 DD 21   E4 1D DB 65   DE 1D 3E FF
1D10 D3 A0 FB 76   76 08 76 FE   20 20 FB 08   CD C6 03 FE
1D20 12 20 21 CD   61 02 04 65   13 14 69 6E   61 14 69 6F
1D30 6E 3A 20 1D   CD DE 03 CD   C6 03 CD FE   1D 3E E7 CD
1D40 57 1E 18 F9   FE 17 20 33   CD 61 02 13   0F 15 12 03
1D50 05 3A 20 1D   CD DE 03 CD   C6 03 CD FE   1D 3E 71 32
1D60 F3 1D 3E 57   D3 A0 FB 76   3E F5 C1 C1   CD 57 1E 18
1D70 EC 1A 2F D3   B8 13 FB F9   76 10 EL FE   66 20 64 CD
1D80 C6 1E 79 32   A3 1D 21 66   C6 66 28 3C   FF 23 10 FB
1D90 06 06 3C 00   23 10 FB 3C   FC 03 66 1A   3C FF 23 10
1DA0 FB 11 01 38   6E 1A 06 66   3C 00 23 10   FB 3C FE 23
1DB0 72 23 3C 00   23 73 23 36   66 03 3C F7   23 06 11 3C
1DC0 00 23 10 FB   3C FB 23 06   00 3C E5 23   10 FB 3C F7
1DD0 23 06 1B 3C   FF 23 10 FB   1C 0D 20 CA   66 66 3C FF
1DE0 23 10 FB C7   FF FF FF DB   1D 2F 12 13   FB F9 7E 18
1DF0 72 F9 1D E7   1D DB 1D 18   F3 DB 65 2F   C9 CD 61 02
1E00 14 12 01 03   0E 3A 20 1D   CD C6 1E DD   71 00 CD C6
1E10 03 CD 61 02   13 05 03 14   0F 12 3A 20   1D CD C6 1E
1E20 DD 71 01 CD   C6 03 CD 61   02 0D 15 0D   02 05 12 20
1E30 0F 06 20 13   05 03 14 0F   12 13 3A 20   1D CD C6 1E
1E40 DD 71 02 CD   B9 1E 3E EB   L3 A0 FB 7E   E6 18 20 F6
1E50 E5 21 00 00   39 D1 C9 32   F3 1D 66 14   3E 77 D5 D3
1E60 A0 FB 7E E6   18 28 34 D1   3E 73 10 F2   11 66 82 CD
1E70 61 02 05 12   12 0F 12 20   61 14 20 14   12 01 03 0B
1E80 20 1D DD 7E   66 CD DA 1E   CD 61 02 20   13 05 03 14
1E90 0F 12 20 1D   DD 7E 61 CD   DA 1E C7 C1   DD 35 02 CA
1EA0 00 06 DD 34   01 DD 7E 61   FE 1E 20 0D   DD 3E 61 61
1EB0 DD 34 66 3E   A3 D3 A6 FB   7E DD 7E 66   2F D3 B8 DD
1EC0 7E 61 2F D3   E0 C9 7E E6   6F 4F 7E FE   20 C8 E6 0F
1ED0 47 79 67 4F   67 07 81 86   18 EF C5 CD   EA 1E C6 30
1EE0 EB 76 23 71   23 77 23 EB   C1 C9 06 30   46 FE 64 38
1EF0 05 6A D6 64   18 F7 1E 6A   38 05 6C D6   6A 18 F7 C9
```

Table 4. System software.

Table 5. Software subroutines.

1DF9 Used in READ and WRITE to convert the typed in track number, sector number and number of sectors from decimal to binary, and then dump them into locations 1DE4 to 6 respectively using the index register. These bytes are then sent to the controller, which is then told to step into this track and, by reading an indent field, verify that the head is over the correct track. Also, by reading the CRC, verifies that the track number has been correctly read and does not match the track register's contents by chance. The data destination/source address is transferred from HL to DE and, by clearing HL and adding SP to it, the contents of the stack pointer register are loaded into HL.

1E57 Used in READ and WRITE. On entering this routine, the A register holds a byte which is dumped at 1DF3 to be used by a $\overline{\text{DRQ}}$ interrupt as the lower part of the interrupt routine address (1DE7 for READ, 1D71 for WRITE, 1DF5 for VERIFY WRITE) 20_{10} is loaded into B and DE, which holds the destination/source address, is saved on the computer stack in case a re-read or -write is necessary. The controller is then instructed to read a sector of data to that and succeeding locations. After the read, at 1E63, a check is made for the correct CRC and for the existence of the track and sector. If no faults have occurred, execution jumps to 1E9B where the saved DE is discarded from the stack into BC and, using indexed operations, the number of sectors byte is decremented. If this operation sets the byte to zero, an exit is made because the READ is complete. Otherwise, the sector and, if necessary, the track number are updated for the next sector to be read, the information is sent to the controller registers and another sector is read. If the operation to read the sector fails, the starting address of the data is popped back off the stack and B is decremented. If this does not reduce it to zero, a re-read is attempted and after 20 attempts execution passes to 1E6C et al and the error message appears.

1EC6 Loads decimal data from the keyboard and converts it to binary in register C.

1EDA Displays the contents of A, converted to decimal, on the v.d.u.

1DE7 The READ interrupt routine, called by a $\overline{\text{DRQ}}$. This routine transfers the byte in the controller's data register to the Z80, inverts it to its true form, stores it in the location pointed to by DE, and increments DE ready for the next byte. The interrupt system in the Z80 is automatically disabled when an interrupt is accepted so that the Z80 can service the interrupting device without interference from the interrupting device itself. Standard service routines usually finish with a re-enabling of the interrupt system and then a return. To ensure that the return will occur, the Z80 does not re-enable the interrupt until it has executed the instruction after the enabling instruction F3. This service routine does not have a return, but it uses this one protected instruction after the F3, F9, to load HL into the stack pointer, SP, register. SP is increased by two when the subroutine 1DF9, which loaded HL with the SP,

ended and the return address was popped off the stack. When the DRQ interrupt was accepted, the current PC (program counter) contents were pushed onto the stack and SP decreased by two, as is normal at the calling of any subroutine. Therefore, for the first DRQ, HL and SP are the same and F9 has no effect. The next byte in the subroutine is a HALT, at which the Z80 stops and waits for the second DRQ which, when it arrives, jumps the execution back to the start of the subroutine and pushes another return address onto the stack. When the data byte is read and the F9 is executed, SP, which decremented when the second interrupt was accepted, is pulled back to where it was before the interrupt occurred. Therefore this, and all future DRQs are demoted from calls to being, effectively, simple jumps to 1DE7. Whilst each return address is written on top of the last as the DRQs progress, the first call from the main program remains unaltered one position further up the stack. When all 128 DRQs have passed and the SP has been pulled back again, the INTRQ interrupt occurs and this, having a conventional return at its end, returns execution to the first popped address, i.e. where the original "read a sector" command was given. This forms a neat method of writing the main program because it makes the controller appear as part of the main processor and, more important, it saves time. There are only 32μs during which data can be transferred from the controller to the memory, and the Z80 made ready for the next interrupt. If the sequence servicing the controller takes longer than this, data will be lost and the controller will halt the reading sequence. A conventional return takes 5½μs and the jump from this returning point to the "wait for a DRQ" point requires a further 6μs. The single F9 instruction only takes 3μs, which achieves the same purpose, but just within the 32μs limit.

1D71 The WRITE interrupt routine called by DRQ. This is similar to the previous routine in that the progress of the stack pointer is arrested by repetitive loading from HL. This routine differs because the first two DRQ-pushed addresses are saved, 1D68 and 1D79 respectively. When the 128 DRQs have occurred, INTRQ causes a jump to the status reading routine after which the return occurs to 1D79 at which a jump pushes execution on the 1D68. Here the other DRQ is popped off the stack and a new vector byte, F5, is placed into the A register ready for the third type of DRQ.

1DF5 When checking a written sector, 1E57 is used as the reading subroutine. Because we are interested in the CRC and not the data on the disc, 1DF5 acts like 1DE7 when handling DRQs, except that it makes no attempt to store the unwanted data and just waits for the INTRQ. When this arrives, 1DF5 returns to the point in the main program where the CRC can be checked to see if the track just written has verified itself.

reading any sector track 77, which does not exist! Note that spaces are required after decimal information – the track, sector and number of sectors, but not after the hexadecimal destination address.

If the Read has worked type RUN 1D00, which should cause the unloaded head to return to track 00. Next type WRITE space, and in response to SOURCE type 0000, for TRACK: 40 space, for SECTOR: 0 space, and for NUMBER OF SECTORS 32 space. The head should move in and write to track 40, step to track 41 and continue writing, so that the first 26 sectors fill track 40 and the final 6 fill sectors 0 to 5 track 41. The write operation is slower because after each sector is written it is read back and checked for errors. As before, up to 20 attempts are made before the operation terminates and the ERROR message occurs. Nevertheless, it should only take a few seconds to record the entire 4K monitor.

Explanations of the software subroutines are given in Table 5. To follow the main program 1D00 to 1DFC, a disassembler such as the one given in a recent computer newsletter is useful. The interrupt mode 2 is set and the I register, which is used (as described in part one) to form the top half of the interrupt addresses, is set at 1D and the IX index register is set at 1DE4. The index register is useful as a pointer to an area of memory because any indexed Z80 instructions, i.e. instructions prefixed by DD, will use a byte in the instruction to say which byte relative to 1DE4 is to be used in the instruction. In this case byte number 00 (i.e. 1DE4 itself) stores the required track number, byte number 01 (1DE5) stores the required sector number and byte 02 (1DE6) stores the number of sectors. The status and data registers are read next (not for their contents) to reset the INTRQ and DRQ interrupt lines if they are active due to the power-on sequence. Note that this unit is designed to operate with the mark III operating system, (see the scientific computer newsletter) which contains these same four bytes, DB,05, DB and 1D. They are executed in the high level so that the MM57109 interrupts are not upset by the disc controlled conditions. 1D0E-14 illustrates the way instructions are sent to the controller. The instruction byte loads into A, in this case a Restore instruction, it is sent to the command register, the interrupt is enabled and the Z80 is halted to wait for the interrupt. When it arrives, in this case a INTRQ, the computer reads the interrupt controller byte F1, adds it to the 1D previously stored in the I register and then reads in the byte at 1DF1 and 1DF2 as the address of the INTRQ subroutine, which is 1DF9. Execution passes to this address when the status register is read and inverted back to a true state.

Re-formatting a disc

As well as reading and writing individual sectors, the controller can read and write whole tracks using the index pulse as the start and finish of the operation. As described in part one, even before use the disc is fully recorded with ident fields and dummy data. If the ident fields become magnetically corrupted, the entire track has to be re-recorded, or re-formatted, before it can be used in the sector mode again. To do this, a block of length 5¼K bytes must be set up in r.a.m. and then recorded en bloc. Assembley of this block requires extra r.a.m. over the basic computer's memory and, given this, the operating system can synthesise the track format. I used a 32Kbyte expansion (referred to in the computer newsletter) and assembled this block at C000. To accommodate other r.a.m. locations, the byte at 1D88 must be altered. After RUN 1D00, type FORMAT space and then the track number, in decimal, to be altered. As

continued on page 57

Table 6 Floppy-disc controllar commands used. The asterisked eddrasses are where modifications to the softwara are made when a track is re-formatted.

Address	Byte	Command	Function
1D0F	FF	00	RESTORE the head to track 00 and clear the track register
1D63	57	A8	Assuming the head is to be loaded against the disc. WRITE a single record of IBM format to the track and sector specified by the raspective registers, using FB as the data mark.
1E47	EB	14	SEEK the track specified by the data register by stepping the difference between it and the contents of the track registar. Then, by reading an Ident Field from the track, verify that it is the correct one.
*1E5D	77	88	Assuming that the head is loaded against the disc, READ a single record of IBM format from the track and sactor specified by the respective registers
1E69	73	8C	As above, but it begins by issuing the HLD, head load, signal and waiting for the HLT signal to become active before proceeding.
1EB4	A3	5C	Load the head against the disc and then STEP IN by one track, updating tha track register. Perform a varify of the track as described above.
*1D63	0B	F4	WRITE TRACK. Starting at tha index pulse, data is written continuously up to the next index pulse.
*1E47	EF	10	On a badly corrupted track, it is not possible to varify the head position after a SEEK, so this version of the command omits it.

Floppy disc system

continued from page 76

explained in Table 6, two controller commands have to be altered, 1D63 to write the complete track, and 1E47 to delete the attempt to verify the head position after the seek operation, as presumably the track is being re-formatted because it is impossible to verify on that track.

Re-formatting is accomplished by proceeding as with a standard Write operation, giving the start of the block as a source and dummy data, say 1, 1 and 1 for the track, sector and number of sectors. The head should move in, load itself and write the track. After this has occured and the head has released, the computer should be manually reset using the Reset key or, preferably, Control Z. □

Table 7. Sequence for a DRQ interrupt. Note that interrupts can be accepted during LD, SP, HL, in which case the HALT will not be executed, but instructions up to and including LD, SP, HL are always executed unless NMI occurs.

(Mode 2 interrupt)	9½ us	
IN A,1D	5½ us	Data read in
CPL	2 us	Data inverted to true
LD (DE) A	3½µs	Stored in memory
INC DE	3 us	Move to next location
EI	2 us	Enable interrupt
LD SP,HL	3 us	Pull back SP
HALT	2 us	Halt for interrupt
	30½ us	

```
0000 31 DF 1F 11    00 80 CD CE    03 20 12 05    01 04 19 1D
0010 CD C4 03 3E    80 D3 00 CD    55 03 FE FC    20 0D 18 12
0020 F5 3E 20 12    13 F1 C9 FF    C3 AA 01 FE    06 20 1A CD
0030 DE 03 36 FF    2C 20 FB C7    F5 DB 03 E6    0F 77 23 F1
0040 FE C9 CD 93    03 CD DE 03    E9 FE 09 20    48 CD DE 03
0050 E5 E7 CD DE    03 C1 C5 A7    ED 42 E5 C1    E7 CD DE 03
0060 D1 EB ED B0    C7 FF E3 29    7E FE 76 20    93 23 E3 DB
0070 01 2F FE 0D    20 02 3E 1F    FE 40 38 02    E6 1F FE 1C
0080 28 E2 FE 1E    CA 00 08 FE    1F 20 05 CD    C4 03 ED 45
0090 12 13 ED 45    FF FE F8 20    5A CD 93 03    CD DE 03 11
00A0 40 80 E5 CD    9F 03 CD 33    03 76 FE 00    28 22 FE 1B
00B0 20 11 CD C4    03 CD D5 00    77 23 FE 1D    20 F7 CD C4
00C0 03 18 E0 CD    F7 03 CD EA    03 77 23 CD    72 03 18 D9
00D0 E1 CD 20 03    C7 76 FE 1B    C0 1B 76 1B    FE 1D 28 F5
00E0 FE 2F 28 08    C6 40 12 13    77 23 13 23    2B 1B 3E 1F
00F0 12 18 E7 FE    FD 20 25 1E    80 21 00 1C    16 00 06 08
0100 7E D3 40 07    10 FB 36 08    7A D3 40 07    10 FB D3 80
0110 CD 36 03 0F    23 14 20 E6    1D 20 DE C7    FE 13 20 1A
0120 CD DE 03 E7    CD E7 03 77    CD 1D 03 CD    45 03 18 F0
0130 FF FF FF FF    FF FF FF FF    FF FF FE 07    20 0B 21 00
0140 80 75 23 7C    FE 88 20 F9    C7 FE 04 20    1E CD DE 03
0150 CD 7F 02 7E    CD 8E 02 23    E7 18 E8 FE    E6 03 CC EC
0160 02 CC EC 02    7D E6 0F 28    CD EC 02 7D    0F 20 5A CD
0170 DE 03 E7 E5    CD DE 03 E7    CD E7 03 47    E7 CD E7 03
0180 4F E3 CD 93    03 11 40 80    7E B8 20 04    71 CD 9F 03
0190 CD 4E 03 23    EB E3 7A BC    20 06 7B BD    20 02 E1 C7
01A0 E3 EB 18 E4    FE 08 CC 17    03 C7 F5 C5    D5 11 00 86
01B0 CD 9F 03 E1    CD 9F 03 E1    CD 9F 03 E1    CD 9F 03 E1
01C0 00 CD 9F 03    E1 CD 9F 03    C7 FE FE 20    32 CD 93 03
0

1D0 CD E7 03 08    E7 CD E7 03    21 00 00 01    00 20 11 40
01E0 80 08 ED B1    E2 00 00 08    ED A1 20 F5    2B 2B CD
01F0 9F 03 CD 4E    03 23 7A FE    88 20 01 C7    F1 18 E2 FE
0200 F1 20 1E CD    DE 03 E7 E5    CD DE 03 CD    54 02 E3 C1
0210 7E CD 60 02    23 7C B8 20    F7 7D B9 20    F3 CD 54 02
0220 C7 FE F2 C2    A4 01 CD DE    03 0E 00 DB    00 E6 01 20
0230 FA CD 36 03    02 DB 00 E6    01 20 F0 CD    36 03 4C 06
0240 08 DB 00 E6    01 81 07 4F    CD 36 03 35    10 F3 79 0F
0250 77 23 18 D5    3E FF 47 D3    FF 10 FA C9    D3 20 07 CD
0260 C5 F5 AF D3    20 CD 36 03    35 F1 06 08    4C C1 C9 CD
0270 36 03 35 10    F7 3E FF D3    20 CD 36 03    02 C9 C5 4F
0280 F0 02 7C CD    8E 02 7D CD    8E 02 CD EC    C1 C6 30 FE
0290 E6 F0 0F 0F    0F 0F CD 9D    02 79 E6 0F    28 05 3E 1F
02A0 3A 38 02 D6    39 FE 1B 30    0E 08 FE 1F    3E 1E CD 01
02B0 CD 01 C3 08    C3 01 03 08    FE 1B 28 05    3E 1E CD 01
02C0 03 08 E5 CE    AC 6F 26 02    7E E1 18 E8    1C 1C 09 18
02D0 35 0F 0B 09    11 12 03 0B    0A 0D 0E 17    10 01 02 13
02E0 04 15 16 07    08 19 0C 1C    11 06 12 14    3E 1C 18 11
02F0 3E 1D CD 01    03 3E 1E CD    01 03 3E 1B    CD 01 03 08
0300 C9 F5 C5 E6    1F 07 07 C6    03 06 08 D3    00 07 CD 36
0310 03 CE 10 F7    C1 F1 C9 CD    DE 03 22 F0    1F 2A F0 1F
0320 11 40 80 CD    9F 03 CD 83    03 CD A9 03    23 CD 72 03
0330 7A FE 88 20    F4 C9 E3 C5    46 E3 E3 E3    E3 E3 E3 10
0340 F8 C1 23 E3    C9 11 08 80    D5 CD C4 03    D1 C9 7B E6
0350 07 C6 E7 18    F9 C5 76 E6    1F 47 11 01    80 CD C4 03
0360 1E 10 78 12    13 76 E6 1F    4F 76 FE 20    20 F8 79 90
0370 C1 C9 E7 7D    E6 03 20 02    E7 E7 7D E6    0F C0 CD C4
0380 03 18 1C 7D    E6 0F C8 4F    E6 0C 0F 81    81 81 47 E7
0390 10 FD C9 D5    11 40 80 E7    7A FE 88 20    FA D1 C9 7C
03A0 CD AA 03 7D    CD AA 03 E7    C9 7E C5 47    E6 F0 07 07
03B0 07 07 CD B9    03 78 E6 0F    C1 FG 0A 38    02 D6 39 C6
03C0 30 12 13 C9    E3 F5 7B E6    3F 28 10 E7    18 F8 E3 F5
03D0 7E 23 FE 1D    28 05 CD 82    00 18 F5 F1    E3 C9 CD E7
03E0 03 67 CD E7    03 6F C9 CD    F6 03 C5 07    07 07 07 47
03F0 CD F6 03 80    C1 C9 76 FE    20 30 02 C6    09 E6 0F C9
```

```
0400 D9 21 0B 1E 7E FE 0C 20 03 AF 18 20 2E 0D 56 14
0410 2E 00 46 23 15 28 0D 23 7E CB 20 B0 CB 20 CB 20
0420 B0 47 18 F0 23 7E FE 05 38 01 04 78 D9 C9 C5 47
0430 DB 03 E6 20 FA 78 D3 60 DB 03 E6 20 28 FA 78 C8
0440 F6 80 D3 60 C1 C9 E3 7E 23 E3 FE FF C8 CD 2E 04
0450 18 F4 23 23 7E FE 20 20 FA 23 C5 CD E6 04 C1 C9
0460 D9 F5 21 F4 1F ED 56 FB CD 46 04 16 2B FF F3 F1
0470 CD 14 07 7D 1E 00 6F E5 C6 0A 6F 11 F6 1F EB 3E
0480 0B 12 13 CB 46 28 03 3E 0C 12 13 2B 2B ED A0 ED
0490 A0 7B E6 F0 C6 09 5F CB 5E 28 03 3E 0C 12 D1 23
04A0 7E 2F E6 0F 06 03 23 06 00 4F ED B0 2F C6 09 4F
04B0 3E 0A 12 13 28 02 ED B0 D9 C9 E6 0F 4F 23 7E FE
04C0 20 CB E6 0F CB 01 81 CB 01 CB 01 81 4F 18 EE FF
04D0 FF FF FF CD 7B 05 CD 46 04 3C 01 02 06 3B FF
04E0 3E 00 CD 60 04 C9 7E 23 0E 0F FE 20 28 F8 FE 1B
04F0 38 53 FE 30 30 19 FE 20 20 03 BE 38 0E C6 0D A7
0500 EA 07 05 FE 09 E6 FB CD 2E 04 C9 0E 0C 7E 23 08
0510 D5 EB 21 00 1E 06 0F CD 20 02 2E 0A 77 1A 23 13
0520 E6 0F FE 0E 20 02 2E 0A 36 0B 23 1A 13 FE 28 30 F0
0530 FE 20 28 43 2E 0A 36 0B DB 08 3E 20 2D 20 E1 36
0540 0C 23 7E 18 30 0A C1 C6 20 BE 20 03 03 18 2D 2B CD
0550 95 07 FE 20 0A C1 FE 80 FE 50 38 02 D6 10 C6 B4
0560 C5 4F 06 07 2E 04 C9 1B 38 07 08 3E 20 CD 2E 04
0570 08 E6 3F CD 2E 04 10 F9 EB D1 AF E5 CD AC 07 06
0580 10 7E 23 CD 2E 04 10 F9 E1 C9 C5 CD 2E 04 DB 03
0590 47 E6 60 EA 8E 05 FE 30 30 04 3E 3F D3 60 DB 03
05A0 E6 20 20 FA 78 E6 40 C1 C9 E5 D5 CD AC 07 E5 11
05B0 00 1E 7D C6 09 6F CD B7 07 E1 06 09 CD B7 07 10
05C0 FB 23 06 06 CD B7 07 10 FB EB 2E 0C 7E FE 30 20
05D0 56 2D 7E FE A0 23 23 7E 20 17 FE 38 30 49 2E 03
05E0 E6 0F 28 3A 47 7E 2B 77 23 23 10 F9 2B 36 2E 18
05F0 2D FE 34 30 32 E6 0F 28 25 47 2E 01 7E 36 2E 23
0600 77 2E 08 7E 23 77 2B 2B 7D FE 00 20 F6 10 F2 2E
0610 01 36 30 23 23 7E FE 2E 20 04 36 30 18 F6 2E 0A
0620 06 05 36 A0 23 10 FB 3E 2E 2D BE 20 FC D1 E3 23
0630 7E FE 30 30 03 2B 18 05 E6 0F 32 E0 1F E3 3A E0
0640 1F 85 FE 09 30 3C 6F 23 E5 7E FE 35 38 2A 2D 7D
0650 FE 01 7E 28 11 FE 2E 28 F5 7E FE 39 28 03 34 18
0660 17 36 30 7E 2B 36 30 00 E8 FE 39 20 F4 36 31 3E 2E 2C BE
0670 20 FC 23 77 2B 36 30 00 E1 3E 0A 95 47 36 20 23
0680 10 FB 2E 00 06 0A 7E FE 20 28 05 12 13 23 10 F6
0690 E3 23 7E FE 2C 00 E3 2B E7 E7 E7 2E 11 AF FB 28 03 E7 10 FD
06A0 25 0A 7E FE 7E FE 05 20
06B0 06 06 03 23 7E 12 13 10 CD 14 07 01
06C0 0A 09 76 FE 2D 20 08 7D E6 F9 6F 3E 0C 77 76 08
06D0 7D E6 F0 6F 08 FE 28 38 16 FE 2E 20 03 79 0E 00
06E0 E6 0F 77 23 76 10 EE 06 01 FE 30 30 F7 18 01 71
06F0 FE 20 28 1E 7D E6 F0 C6 0A 6F 36 0B 23 06 02 76
0700 FE 2D 20 05 3E 0C 77 18 F4 23 00 E6 0F 77 00 10
0710 EE E7 E1 C9 CD AC 07 06 09 AF 77 23 10 FC 06 06
0720 3E 0F 77 23 10 FC 36 3F C9 76 47 76 4F 76 FE 20
0730 20 FA 79 80 80 C9 76 E6 0F 4F 76 FE 20 C8 E6 0F
0740 47 79 27 4F 07 07 81 80 13 EF C5 06 30 FE 64 38
0750 05 04 D6 64 18 F7 0E 30 FE 0A 38 05 0C D6 0A 18
0760 F7 C6 30 EB 70 23 71 23 77 23 EB C1 C9 C5 01 30
0770 30 FE 64 38 05 D6 64 04 18 F7 FE 0A 38 05 D6 0A
0782 0C 18 F7 F5 78 CD A5 02 79 CD A5 02 F1 C6 30 CD
0790 A5 02 C1 C9 23 4E 23 4E 23 7E FE 20 20 F9 79 80
07A0 80 C9 17 12 09 14 05 10 12 09 0E 14 A7 26 1E 17
07B0 17 17 17 6F D0 24 C9 7E FE 0B 20 02 3E 0D 5E 0A
07C0 20 02 3E FE 3E 70 FE 0C 20 02 3E FD
07D0 C6 30 12 13 23 C9 31 0C 3F 3F 35 37 2C 00 3F 3F
07E0 1D 1C 3F 22 23 3F 3F A5 24 3F 26 33 34 25 2D 2F
07F0 2B 1B 21 36 32 3F 3F A4 3F A6 3F 3F 3F 30 38 F
```

```
0800 31 DF 1F 11 00 80 CD CE C3 20 02 15 12 10 20 1D
0810 E7 E7 D5 CD C4 03 D1 3E 0E 32 E0 1F 1E 08 3E 40
0820 CD 36 03 10 3E 3F D3 60 ED 56 CD 36 03 90 3E BF
0830 D3 60 CD 46 04 2F 22 FF CD 29 07 FE 14 20 7F E7
0840 CD 36 07 21 00 0C 7E FE CD 28 B5 23 FE ED 20 F6
0850 7E B9 20 F3 2B 2B E5 D1 13 1A FE 1F 20 FA 1A 77
0860 23 13 7A FE 1E 2B F7 0E 01 CD 93 03 11 40 80 0D
0870 0C 20 05 CD 00 03 18 88 21 00 0C 7E FF 00 28 F0
0880 FE ED 23 2B F6 7E B9 20 F7 CD 4A 07 E7 7A FE 88
0890 20 16 CD CE 14 20 09 0E 03 0C 09 13 23 7E FE C0 28 C2 12 13
08A0 0C 05 14 05 1D 76 18 C1 23 7E FE 0E 20 B4 18 B6 F5 CD
08B0 FE 1F 20 F4 CD C4 03 7A FE 0B 23 7E FE C0 28 97
08C0 93 03 F1 FE 1E 20 2C 21 FE 3A 20 0D 11 A6 07 01
08D0 FE ED 20 F6 23 CD 94 07 E1 FE 33 20 DD 11 AB 07
08E0 05 00 EB 1B ED B8 EB 18 FF 0B 23 7E FE C0 20 FA
08F0 18 ED 0E FE 06 20 30 21 36 ED 23 CD 36 07 71 23
0900 18 03 21 00 0C 11 40 80 77 23 FE 1F 28 EA 18 F0
0910 76 FE 3A 28 0B CD D6 00 1C 28 D7 FE 2C 20 06 CD
0920 36 1F 23 36 C0 18 95 FE 3A CD 36 07 CD F0 02 0D
0930 36 07 C3 69 08 FE 18 20 21 00 0C 7E FE C0 28 F0
0940 0C 20 05 CD 7F 02 18 DD F7 CD 6D 07 CD EC 02 23
0950 FE ED 23 20 F6 7E B9 20 05 CD A5 02 18 F1 CD F0
0960 7E FE C0 28 DB FE 1F 28 36 07 11 40 80 0D C5 C1
0970 02 18 CD FE 32 20 BE CD 28 F5 FE ED 23 20 F6 7E
0980 0C C5 21 00 0C 7E FE C0 94 07 FE 26 20 14 CD C4 03 23 7E CD
0990 B9 20 F7 CD 94 07 FE 26 FE 1F 20 F2 18 D1 FE 34
09A0 BB 06 23 7E FE 20 28 FA 03 23 7E FE 20 28 FA FE 1F 28 C0 FE
09B0 20 22 CD C4 03 23 7E FE EC 12 13 18 F6 FE 1B DC
09C0 22 20 0A 23 7E FE 22 28 C3 00 08 FE 1D 20 08 23
09D0 A9 05 18 E1 FE 0E 20 03 2C 20 15 23 46 C5 23 23
09E0 7E CD BA 04 E1 18 9A FE 04 18 84 CD E6 04 18 EF
09F0 7E FE 1F 20 06 F1 CD 60 05 23 46 23 7E C5 CD E6
0A00 FE 18 20 3B 23 7E CD 7B 3E 30 CD 2E 04 3E 3A CD
0A10 04 C1 78 A7 FE 3E 20 05 3E 11 18 02 3E 12 CD 8A
0A20 2E 04 78 A7 FE 3D 20 04 FA 18 A6 18 BC 18 A5 FE
0A30 05 20 C6 23 7E FE 30 38 C0 1D D5 23 7E FE 20 28
0A40 33 20 3C CD F0 02 D5 11 0A 23 7E FE 22 28 EC 12
0A50 FA FE 1F 28 15 FE 22 20 18 E1 3E FF 12 00 D1 1A
0A60 13 18 F6 FE 1B DC A9 05 02 13 18 F3 D1 18 BC FE
0A70 FE FF 28 08 E6 3F CD A5 03 18 F2 FE 38 20 0B 11
0A80 0F 20 08 11 40 80 CD 93 18 EF FE 10 20 08 23 7E
0A90 40 80 D5 E7 CD C4 03 D1 20 04 C1 00 18 8D FE 1E
0AA0 CD BA 04 E5 18 97 FE 32 1F 23 46 23 CD 59 04 78
0AB0 20 2C C1 C5 0C 79 32 E3 52 04 3E 1B CD 60 04 CD 46 04 30 FF
0AC0 CD 60 04 CD 52 04 CD 30 3C 1C FF 18 9F FE 30 28 EE 23 7E
0AD0 CD 52 04 CD 46 04 3D 3A 13 CD 8A 05 28 EE 23 7E
0AE0 20 2D CD 46 04 1D FF 3E 05 CD 46 04 39 FF F1 CD
0AF0 F5 CD 7B 05 3E 1B CD 7B FF 3A E3 1F 4F 18 95 FE
0B00 60 04 CD 46 04 01 20 3A 80 21 E4 1F D5 E5 06 02
0B10 24 20 01 76 18 C6 11 40 E1 01 04 00 ED B0 EB D1
0B20 ED 5F CD AB 0B 10 F9 EB 10 F8 E7 CD A0 0B 1A BE
0B30 06 02 CD E7 03 CD AB 0B 23 10 F3 79 D9 FE 34 28
0B40 20 05 2F 12 36 EE 0C 13 0C 01 03 0B 13 2C 20 1D
0B50 46 12 13 CD CE 03 20 02 0E BE 20 05 0C 36 DD 2F
0B60 CD A0 0B EB 1A FE 10 30 2E E3 10 E8 79 D9 12 13
0B70 12 23 7D FE EC 20 ED 13 05 13 1D CD C4 03 21 E4
0B80 CD CE 03 20 17 08 09 14 CE 03 19 05 13 21 1D C7
0B90 1F D5 E5 2E E8 18 90 CD EC 1F C9 E6 77 77 AF ED
0BA0 D9 01 30 04 11 E8 1F 21 0F 12 20 01 3D 31 20 13
0BB0 67 23 77 23 C9 ED 05 06 14 09 0C 20 32 35 20 1F
0BC0 14 05 10 20 31 20 15 0E 01 20 0C 0F 07 20 1F ED
0BD0 ED 06 0C 05 14 20 0C 3D 30 20 0C 38 20 1F ED 08
0BE0 07 10 12 09 0E 14 20 01 ED 09 05 0E 04 20 1F C0
0BF0 0E 05 18 14 20 01 20 1F
```

```
0B00  60 04 CD 46   04 01 20 3A   FF 3A E3 1F   4F 18 95 FE
0B10  24 20 03 76   18 C6 FE 15   20 14 23 CD   E6 04 3E 10
0B20  CD 60 04 23   CD E6 04 3E   11 CD 60 04   18 E6 FE 16
0B30  20 FA 23 CD   E6 04 CD 46   04 02 3C FF   3E 10 CD D4
0B40  04 CD 00 04   F5 23 CD E6   04 3E 11 CD   D4 04 CD 00
0B50  04 D9 57 F1   5F 0E 08 7A   E6 01 28 02   CB 39 7B E6
0B60  01 28 06 CB   39 CB 39 CB   39 79 A7 20   02 3E 02 C6
0B70  40 4F CB 22   CB 3B 3E 1F   93 5F 06 02   CB 2B CB 1A
0B80  10 FA 3E 80   83 5A 57 79   12 D9 18 A0   FF FF FF FF
0B90  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF
0BA0  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF
0BB0  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF
0BC0  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF
0BD0  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF
0BE0  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF
0BF0  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF
```

THESE ARE THE CHANGES REQUIRED IN THE LAST SECTOR OF THE THIRD 2708 IN ORDER FOR THE GRAPH PLOTTING INSTRUCTIONS DESCRIBED IN PART 5 OF THE SERIES OF ARTICLES, TO WORK. THE BLOCK OF FF'S FROM 0B3C TO 0BFF ARE JUST BLANK SPACES, AVAILABLE FOR LATER DEVELOPMENT.